日　本　国　特　許　庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　３月　７日

出願番号
Application Number: 特願２００１－０６２７９２

出願人
Applicant(s): 株式会社日立製作所

U.S. Appln. Filed 12-20-01
Inventor: F. Arakawa
Mattingly Stanger & Malur
Docket NIT-311

２００１年１２月　７日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３１０６８０６

【書類名】　特許願

【整理番号】　H01001351A

【あて先】　特許庁長官　殿

【国際特許分類】　G06F　9/38

【発明者】

【住所又は居所】　東京都国分寺市東恋ケ窪一丁目２８０番地　株式会社日立製作所中央研究所内

【氏名】　荒川　文男

【特許出願人】

【識別番号】　000005108

【氏名又は名称】　株式会社　日立製作所

【代理人】

【識別番号】　100075096

【弁理士】

【氏名又は名称】　作田　康夫

【電話番号】　03-3212-1111

【手数料の表示】

【予納台帳番号】　013088

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】明細書

【発明の名称】スレッド間優先度可変プロセッサ

【特許請求の範囲】

【請求項１】

複数のプログラムカウンタと、１つまたは複数の命令実行部と、該複数プログラムカウンタのそれぞれに対応する複数スレッドの命令フローを該命令実行部に選択的に供給する手段を有し、該複数スレッドを同時または時分割に実行可能なマルチスレッドプロセッサにおいて、該複数スレッド間に時分割に変更可能な実行優先度があって該優先度に従って逐次処理した場合と同一の結果を生成することを特徴とするプロセッサ。

【請求項２】

請求項１記載のプロセッサにおいて、複数スレッド間でプログラムカウンタ以外のプロセッサリソースの一部または全部を共有することにより、ハードウェア量の削減と、共有リソースによるスレッド間のデータ受け渡しを可能にしたことを特徴とするプロセッサ。

【請求項３】

請求項１または２記載のプロセッサにおいて、繰り返し回数を第１の優先度判定基準、複数スレッド間の優先度を第２の優先度判定基準とすることにより、命令の介在なしにハードウェアが複数スレッド間の同期をとることを可能にしたプロセッサ。

【請求項４】

請求項１から３記載のプロセッサにおいて、最優先でないスレッドの実行結果を一時的に保持するバッファを有し、より優先度の高い処理の終了または同期報告後に、本来の格納場所に格納することにより、最優先でないスレッドの実行を矛盾なく行うことを可能にしたプロセッサ。

【請求項５】

請求項１から４記載のプロセッサにおいて、複数スレッド間のデータ依存関係を、データの流れが単一方向となるように限定し、データ使用スレッドを最優先スレッドの場合のみ実行することにより、定義前データの使用をなくして、最優先

でないスレッドの実行を矛盾なく行うことを可能にしたプロセッサ。

【請求項６】

請求項１から５記載のプロセッサにおいて、スレッド間データ通信に用いるデータ格納場所を限定したことを特徴とするプロセッサ。

【請求項７】

請求項６記載のプロセッサにおいて、該データ格納場所をスレッドの組合せ及び通信方向毎に独立に複数箇所定義したことを特徴とするプロセッサ。

【請求項８】

請求項７記載のプロセッサにおいて、該複数のデータ格納場所毎に実行優先度を定義したことを特徴とするプロセッサ。

【請求項９】

請求項６から８記載のプロセッサにおいて、該データ格納場所がレジスタ又はメモリの一部であることを特徴とするプロセッサ。

【請求項１０】

請求項１から９記載のプロセッサにおいて、優先度の低いスレッドへの優先スレッド化命令を有し、複数スレッド間の優先度変更を容易にしたことを特徴とするプロセッサ。

【請求項１１】

請求項１から７記載のプロセッサにおいて、他のスレッドへのデータ定義同期命令を有し、同期後の他スレッドのデータ使用を可能にしたプロセッサ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はマイクロプロセッサ等のデータ処理装置にかかり、マルチスレッドプロセッサのスレッド管理をするための有効な手段を提供する。該マルチスレッドプロセッサは、オペレーティングシステム等のソフトウェアの介在なしに、複数のスレッドを時分割または同時に実行することが可能なプロセッサである。該スレッドは、少なくとも固有のプログラムカウンタを有する命令フローであり、複数スレッド間でのレジスタファイルの共有が可能なスレッドとする。

【０００２】

【従来の技術】

逐次処理フローを逐次処理より実行並列度を上げて高速に実行する方法には様々な方法がある。（１）依存関係のない複数処理を一つの命令にまとめて同時実行するSIMD(Single Instruction Multiple Data)命令やVLIW(Very Long Instruction Word)命令、（２）依存関係のない複数命令を同時に実行するスーパスカラ方式、（３）命令単位で、逐次処理フローの順序と異なる順序で実行して、データ依存関係やリソース競合による実行並列度低下やストールを削減するアウトオブオーダ実行方式、（４）自然な逐次処理フローの順序を実行並列度が最も上がるようにあらかじめ並べ替えたプログラムを実行するソフトウェアパイプラインニング、（５）逐次処理フローを複数命令から成る複数命令列に分割して、この複数命令列をマルチプロセッサまたはマルチスレッドプロセッサで実行する方式等である。（１）（２）は並列処理のための基本方式、（３）（４）は局所的な並列度をより多く抽出するための方式、（５）は大局的な並列度を抽出するための方式である。

「MICROPROCESSOR REPORT,vol.13,no.13, Oct.6, 1999, pp.1,6-10」記載のIntel社のMercedは上記（１）のVLIW方式を実装し、（４）のソフトウェアパイプラインニング方式のために整数128本、浮動小数点128本の計256本の64ビットレジスタを実装している。レジスタ本数が多いため数十命令規模の並列度抽出が可能である。

「MICROPROCESSOR REPORT,vol.13,no.16, Dec.6, 1999, pp.1,6-11」記載のCompaq社のAlpha 21464は、上記（２）のスーパスカラ方式、（３）のアウトオブオーダ方式、及び（５）のマルチスレッド方式を実装している。大容量の命令バッファ及びリオーダバッファによるアウトオブオーダ実行で数十命令規模の並列度を抽出し、マルチスレッド方式によって更に大局的な並列度を抽出し、スーパスカラ方式で並列実行している。したがって、全般的な並列度抽出が可能であると考えられる。但し、複数スレッド間の依存関係解析を行わないので、依存関係のある複数スレッドの同時実行はできない。

「MICROPROCESSOR REPORT,vol.14,n .3, March, 2000, pp.14-15」記載のNEC社

のMerlotは（５）のマルチプロセッサの例である。Merlotは密結合オンチップ４並列プロセッサで複数スレッドの同時実行を行っている。依存関係のある複数スレッドの同時実行も可能である。依存関係解析を容易にするために、新スレッドは最新の既存スレッドのみが生成し、逐次実行した場合の順序は新スレッドが最後であるという制約を付けている。

「特開平8-249183 推論並列命令スレッドの実行」記載のCPU（中央演算処理装置）は（５）のマルチスレッドプロセッサの例である。メインスレッドと将来スレッドを同時実行するマルチスレッドプロセッサである。メインスレッドは逐次実行のためのスレッド、将来スレッドは逐次実行した場合に将来実行するプログラムを投機的に実行するためのスレッドである。将来スレッドの使用するレジスタまたはメモリ上のデータは将来スレッド開始時のデータであり、逐次実行した場合の将来スレッド開始時点までに更新される可能性がある。更新されれば将来スレッドの使用したデータは正しくないので将来スレッドの結果を破棄し、更新されなければ残す。更新の有無は、逐次実行した場合に将来スレッド開始時点に至るまでのプログラムフローを条件分岐の方向によってチェックし、更新命令を実行するフローかどうかで判断する。このため、複数スレッド間の依存関係解析が不要であるという特徴を持つ。

【０００３】

【発明が解決しようとする課題】

例えば、図１のプログラムは８個のデータを加算するプログラムである。本プログラムを実行するプロセッサは図２のようなリピート制御命令を持っているものとする。これらの命令でリピート実行前にリピート構造を構成すると、リピートカウンタ更新命令、リピートカウンタチェック命令、及び条件分岐命令等のリピート制御命令をリピート中に実行する必要がないものとする。このようなリピート制御命令はデジタルシグナルプロセッサ（DSP）では一般的であり、汎用プロセッサにも容易に適用可能である。

本プログラムを図３に示すパイプライン構成のロードレイテンシ4の２並列スーパスカラプロセッサで実行する場合を考える。図中、Iは命令フェッチ、D0、D1は命令デコード、Eは加算、ストア等の実行、L0-L3はロードステージである。パ

イプライン動作は図４のようになる。図中、命令#7はレジスタr0のアドレスからデータをレジスタr2にロードし、レジスタr0を次のアドレスに更新する命令である。命令デコードステージD0でデコードされ、4サイクルのロードステージL0-L3でロードが実行されてL3ステージの最後でロードデータが使用可能となる。同時にL0ステージでアドレス更新が実行されてL0ステージの最後で更新アドレスが使用可能となる。一方、命令#8はレジスタr2とレジスタr3を加算し、結果をレジスタr3に格納する命令である。命令デコードステージD1でデコードされ、実行ステージEで加算が実行されてEステージの最後で結果が使用可能となる。命令#8は命令#7のロード結果を使用するため、命令#7のL3ステージの次サイクルでEステージを実行する。そして、ロードレイテンシが隠蔽できないためにN個のデータの加算に４N+2サイクルかかる。ロードレイテンシをLとするとLN+2サイクルである。更に、外部メモリアクセスを想定してロードレイテンシを例えば30と設定すると、N個のデータの加算に30N+2サイクルかかってしまう。

次に上記プロセッサに上記Alpha 21464のようなアウトオブオーダ実行機能を付加すると、ロードレイテンシ4ならば図５のように動作しN+5サイクル、30ならばN+31サイクル、LならばN+L+1サイクルで完了する。しかしながら、ロードレイテンシ30に対応するには60命令レベルの並べ替えが必要である。図１のプログラムでNを30以上に設定したとすると、60命令の内、30のADD命令を命令バッファに保持しながら、30のロード命令を実行していく。そして、ADD命令実行後に本来の実行順序で実行結果を書き戻す。このため、Alpha 21464のような大容量の命令バッファ及びリオーダバッファが必要となり、プロセッサのコストパフォーマンスが低下する。

次に、図1のプログラムを上記Mercedのようなソフトウェアパイプラインニング方式で高速化すると、ロードレイテンシ4ならばプログラムは図６のようになる。そして、パイプラインは図７のようになり、上記アウトオブオーダ実行同様N+5サイクルで完了する。この時、図１のプログラムに比べて３本のレジスタを余分に使用する。そして、ロードレイテンシ30に対応するには29本の余分なレジスタを用いたプログラムに変更しなければならない。実行サイクルはN+31である。このようにソフトウェアパイプラインニング方式では多数のレジスタとレイテン

シに合せた最適化が必要である。一般には、プログラムが仮定したロードレイテンシをX、実際のロードレイテンシをLとすると実行サイクル数はMAX(1,L-X+1)N+MAX(L,X)+1サイクルとなる。MAX(式1,式2)という関数は、最大値選択関数であり、式1及び式2の大きい方を選択する。過小なレイテンシを仮定すると第1項が増えてしまい、過大なレイテンシを仮定すると第2項が増えてしまう上にレジスタを浪費する。外部メモリアクセスレイテンシは動作周波数を変えただけでも変化するのでソフトウェアの汎用性は低い。また、通常の32ビット命令のプロセッサはレジスタが32本なのでレジスタ本数が不足する。

以上のように上記Alpha 21464及びMercedの方式は数十命令レベルの並列度抽出による高速化は可能であるが、コストパフォーマンスが低かったり、通常の32ビット命令では対応できなかったりするため、高価なプロセッサでなければ採用できない。

一方、上記Merlot用に図1のプログラムを変更すると図８のようになる。パイプラインは図９のようになり、新スレッド発行がネックとなって、N個のデータの加算に2N+7サイクルかかる。一つのプロセッサに着目すると４スレッド毎に１スレッドの処理を担当し、１スレッドの処理に７サイクルかかる。ロードレイテンシがLならばL＋３サイクルかかる。一方、新スレッド発行は２サイクルピッチなので、２×４＝８サイクル毎に同一プロセッサに新スレッドを発行できる。同一プロセッサで実行されるスレッドは逐次実行されるため、処理時間L+3と発行ピッチ8の大きい方に実行時間が律速され、N個のデータの加算にMAX(L+3,8)N/4+7サイクルかかる。ロードレイテンシが30ならば33N/4+7サイクルかかる。2並列スーパスカラプロセッサを４個搭載している割には性能が低い。

最後に上記特開平8-249183のマルチスレッドプロセッサ方式用に図1のプログラムを変更すると図10のようになる。将来スレッド発行／終了に命令が必要なため実際の処理の２命令と合せて１個のデータに対して４命令必要である。また、将来スレッド発行後に将来スレッドとして実行されたコードに必ずメインスレッドが到達しなければならない。到達時に将来スレッドの実行結果を採用するか破棄するか決定するからである。次のリピート処理のために将来スレッドを発行したらリピートを抜けてしまって、メインスレッドが次のリピート処理をしなかった

という事態は避けなければならない。したがって、リピートの先頭でリピートの最後にある将来スレッドを発行する場合が最も早い将来スレッド発行である。この結果、将来スレッド発行ネックとなり、２並列スーパスカラプロセッサ方式では図11のようにN個のデータの加算に3N+5サイクルかかる。この時、図11の#10のADDとその3命令後の#9のFORKが同時に実行されている。そして、ロードレイテンシが30になると、これら#10と#9の実行は図11より26サイクル後になる。この結果、サイクル数はロードレイテンシで決まり29N+5サイクルである。一般にはMAX(3N+5,(L-1)N+L+1)サイクルである。上記Alpha 21464、Merced及びMerlot方式に比べてハード量は少ないが性能も低い。

以上を纏めると図12のようになる。#1はデータ数N、ロードレイテンシLの場合に一般化した場合、#2はロードレイテンシが４と比較的短い場合、#3はロードレイテンシが30と比較的長い場合、#4～#7は具体的なデータ数、ロードレイテンシの場合である。特に、ロードレイテンシが長い場合に、既存のマルチスレッドプロセッサでは並列性抽出が困難であることが分かる。

本発明が解決しようとする課題は、Alpha 21464のような大規模なハードウェア追加やMercedのような根本的なアーキテクチャ変更を行わずに、少ないハードウェアの追加でAlpha 21464やMerced並みの数十命令規模の並列度抽出を可能にし、性能を向上させることである。特に、単一プロセッサで複数スレッドを実行するマルチスレッドプロセッサを改良することにより、数十命令規模の並列度抽出を可能にすることである。

【０００４】

【課題を解決するための手段】

従来のマルチスレッドプロセッサは複数スレッドに逐次実行順序を付与することにより、新スレッド発行及び依存関係解析を単純化している。しかしながら、この方式では図１のような単純なプログラムであっても、並列性抽出が困難である。本発明ではこれらの制約を効果的に解消することにより、数十命令規模の並列度抽出を可能にする。

従来のマルチスレッドプロセッサが新スレッドに固定の逐次実行順序を付与するのに対し、本発明ではスレッド実行中の逐次実行順序変更を可能にする。これに

よって、本発明では従来方式とは異なるスレッド分割が可能である。図13はスレッド分割の違いの模式図である。図中の各命令の番号は逐次実行順序である。番号の小さいもの程早く、#00,#01,#10，#11,･･･,#71の順序となる。従来方式においては、逐次処理を単純に時分割してスレッドを割り当てている。このため、先行実行したい処理数のスレッドを生成しなければならない。図13では8個のスレッドに分割し、新スレッド発行命令FORKで発行している。図示していないがスレッド終了命令も必要となる。そして、スレッド生成数に制限があれば、それが先行処理数を制限する。本発明では先行処理と他の処理にスレッドを割り当て、これら２つの処理を逐次実行順序の時分割変更を行いながら実行する。２つのスレッドで多数の先行処理が可能である。図13のSYNCが逐次実行順序変更点である。例えば、TH0の命令#00と#10の間及びTH1の命令#01と#11の間に逐次実行順序変更点SYNCがあるので、逐次実行順序変更点SYNCより前の命令#00及び#01は、TH0の命令#10以降及びTH1の命令#11以降より逐次実行順序が早い。以下同様にして逐次実行順序が与えられる。逐次実行順序変更点SYNCは命令によって指定することができる。また、図2のリピート制御命令によってリピート構造を定義した場合は、リピート終了PCからリピート開始PCに戻る時点を逐次実行順序変更点SYNCとすれば特別な命令は不要である。

図14はロードレイテンシ８の場合の従来方式のスレッド実行状態である。本発明との比較を容易にするためにFORK命令を毎サイクル発行できるものとする。最高の性能を得るためには８スレッドが同時に存在しなければならない。レイテンシが30ならば30スレッド必要である。図15はロードレイテンシ８の場合の本発明のスレッド実行状態である。２スレッドのみで最高の性能を得られる。レイテンシが30になっても２スレッドで済む。また、逐次実行順序変更は命令に付与する内部状態を変更するだけなので、新スレッド発行命令FORKより容易であり、簡単なハードで毎サイクル実行できる。

依存関係にはフロー依存、逆依存、出力依存の3種類がある。同一のレジスタやメモリアドレスに対するアクセスに関して、フロー依存は「読み出しは先行する全書込みの終了後に行う」、逆依存は「書込みは先行する全読み出しの終了後に行う」、出力依存は「書込みは先行する全書込みの終了後に行う」という依存関

係である。これらの規則を守れば命令の実行順序を入れ替えても、入れ替えない場合と同一の結果が得られる。

上記依存関係のうち、逆依存および出力依存は、異なるデータの格納場所を時分割で同一のレジスタやメモリアドレス上に確保するために起こる依存関係である。このため、一時的なデータ格納場所を確保して、格納場所を分離することによって回避すれば、逆依存および出力依存があっても逐次実行順序の遅いスレッドの実行開始は可能である。本発明も従来のマルチスレッドプロセッサでもこの方式を取っている。

一方、フロー依存の規則は守らなければならない。従来のマルチスレッドプロセッサでは、命令実行時にフロー依存の有無が判明しない場合は実行結果を一時的なデータ格納場所に残し、フロー依存がないことが判明したら正規の格納場所に格納し、フロー依存があることが判明したら処理をキャンセルして再実行して正しい結果を得る。しかし、この方式では正常動作はするが、高速動作は保証されない。

本発明では、キャンセル再実行という可能性をなくすことにより確実に高速動作させる。マルチスレッドプロセッサがフロー依存解析に失敗する理由は、データ定義命令デコード前にデータ使用命令をデコードして実行してしまう場合があるからである。本発明では、定義命令が必ず先にデコードされるように制約を加える。尚、アウトオブオーダ実行方式では、実行はアウトオブオーダであるがデコードはインオーダであるため、この問題は生じない。その代わり実行数以上の多数の命令をデコードして実行できる命令を選択し実行部に供給しなければならない。

図13のような本発明のスレッド分割方式では一方のスレッドがデータを定義し、他方がデータを使用する。そこで、データ定義スレッドとデータ使用スレッドを定義し、データ定義スレッドがデータ使用スレッドのデータを使用することを禁止する。即ちデータの流れをデータ定義スレッドからデータ使用スレッドへの一方通行とする。そして、データ定義スレッドはデータ使用スレッドを追い越して良いが、データ使用スレッドはデータ定義スレッドを追い越してはならないと定義する。こうすると、データ定義スレッドはデータ使用スレッドに対するフロー

依存解析が不要なので追い越しても誤動作せず、データ使用スレッドはデータ定義スレッドを追い越さないのでフロー依存解析を誤ることはない。
本発明用に図1のプログラムを変更すると図16のようになる。命令#1,#3,及び#7によって命令#9のリピート構造を、命令#11-#13によって命令#15のリピート構造を定義している。そして、リピートタイプのスレッド生成命令THRDG/Rによって第2スレッドを起動することにより、リピート終了PCからリピート開始PCに戻る時点を逐次実行順序変更点SYNCとして２つのスレッドのリピート構造を構成することができる。スレッド生成命令THRDG/Rを発行したスレッドがデータ定義スレッド、スレッド生成命令THRDG/Rによって生成されたスレッドがデータ使用スレッドである。
本発明を適用したプロセッサを図17のようなロードレイテンシ4のパイプライン構成と仮定する。命令アドレスステージA0及びA1はパイプラインとして明示しないのが慣例であり、従来例の説明では省略していたが、本発明の動作を説明するために明示する。この時、パイプライン動作は図18のようになり、実行サイクル数はN+5である。そして、レイテンシを30にするとN+31、Lとするとと N+L+1となる。すなわち、大規模なアウトオブオーダ実行やソフトウェアパイプラインニングを行った場合と同等な性能である。尚、図18のパイプライン動作は、具体的な実施例を用いて詳細に後述する。

【０００５】

【発明の実施の形態】

図19は本発明を適用した2スレッドプロセッサの例である。命令供給部IF0、IF1、命令アドレスマルチプレクサMIA、命令マルチプレクサMX0、MX1、命令デコーダDEC0、DEC1、レジスタスコアボードRS、レジスタモジュールRM、命令実行部EX0、EX1、メモリ制御部MCから成る。以下に各部の動作を示す。尚、本発明の重要モジュールである命令供給部IF0、IF1、命令マルチプレクサMX0、MX1、レジスタスコアボードRS、及びレジスタモジュールRMについては、詳細動作を後述する。また、本実施例においては説明を容易にするために命令供給部IF0がデータ定義スレッド、命令供給部IF1がデータ使用スレッド用に固定されているものとする。この固定をはずすことは本発明の属する技術分野の通常の技術者ならば可能で

ある。また、命令マルチプレクサMX0、命令デコーダDEC0、及び命令実行部EX0をパイプ0、MX1、DEC1、及びEX1をパイプ1とする。

命令供給部IF0又はIF1は命令アドレスマルチプレクサMIAにそれぞれ命令アドレスIA0、IA1を出力する。命令アドレスマルチプレクサMIAは命令アドレスIA0及びIA1の一方を命令アドレスIAとして選択してメモリ制御部MCへ出力する。メモリ制御部MCは命令アドレスIAから命令をフェッチし、命令ILとして命令供給部IF0又はIF1に出力する。命令供給部IF0とIF1は同時に命令フェッチすることはできないが、1度にフェッチする命令数を２命令以上にすれば命令フェッチがボトルネックとなることはまれである。命令供給部IF0はフェッチした命令のうち、先頭2命令をI00及びI01としてそれぞれ命令マルチプレクサMX0及びMX1に供給する。同様に命令供給部IF1はフェッチした命令のうち、先頭2命令をI10及びI11としてそれぞれ命令マルチプレクサMX1及びMX0に供給する。

また、命令供給部IF1はスレッドが2本走っている場合のみ動作する。スレッドが1本から2本になる場合には、命令供給部IF0から命令供給部IF1及びレジスタスコアボードRSへのスレッド生成GT0をアサートし、命令供給部IF1が起動する。スレッドが1本に戻る場合は命令供給部IF1がスレッド終了ETH1をアサートし、停止する。

命令マルチプレクサMX0は、命令I00及びI11から命令を選択し、命令コードMI0を命令デコーダDEC0へ、レジスタ情報MR0をレジスタスコアボードRSへ出力する。

同様に、命令マルチプレクサMX1は、命令I10及びI01から命令を選択し、命令コードMI1を命令デコーダDEC1へ、レジスタ情報MR1をレジスタスコアボードRSへ出力する。

命令デコーダDEC0は、命令コードMI0をデコードし、命令実行部EX0へ制御情報C0を、レジスタスコアボードRSへレジスタ情報有効VR0を出力する。レジスタ情報有効VR0はRA0及びRB0の読出し、RA0及びRB0への書込みのそれぞれについて有効を表すVA0,VB0,V0,及びLV0から成る。同様に、命令デコーダDEC1は、命令コードMI1をデコードし、命令実行部EX1へ制御情報C1を、レジスタスコアボードRSへレジスタ情報有効VR1を出力する。レジスタ情報有効VR1はRA1及びRB1の読出し、RA1及びRB1への書込みのそれぞれについて有効を表すVA1,VB1,V1,及びLV1から成る

。

レジスタスコアボードRSはレジスタ情報MR0及びMR1、レジスタ情報有効VR0及びVR1、並びに、スレッド生成GTH0及びスレッド終了ETH1から、レジスタモジュール制御信号CR、並びに、命令マルチプレクサ制御信号CMを生成し、それぞれレジスタモジュールRM、並びに命令マルチプレクサMX0及びMX1へ出力する。

レジスタモジュールRMは、レジスタモジュール制御信号CRに従って、命令実行部EX0への入力データDRA0及びDRB0、並びに、EX1への入力データDRA1及びDRB1を生成し、それぞれ命令実行部EX0並びにEX1へ出力する。また、命令実行部EX0及びEX1からの演算結果DE0及びDE1、並びに、メモリ制御部MCからのロードデータDL3を格納する。

命令実行部EX0は、制御情報C0に従って入力データDRA0及びDRB0を処理し、実行結果DE0をメモリ制御部MC及びレジスタモジュールRMに、実行結果DM0をメモリ制御部MCに出力する。同様に、命令実行部E1は、制御情報C1に従って入力データDRA1及びDRB1を処理し、実行結果DE1をメモリ制御部MC及びレジスタモジュールRMに、実行結果DM1をメモリ制御部MCに出力する。

メモリ制御部MCは、命令実行部EX0又はEX1で処理された命令がメモリアクセス命令であった場合に実行結果DE0又はDE1を使用してメモリアクセスを行う。この時、アドレスAを出力し、データDをロードまたはストアする。更に、メモリアクセスがロードであった場合は、ロードデータDL3をレジスタモジュールRMに出力する。

図17のパイプラインに対応させると、命令供給部IF0及びIF1の命令アドレス系動作が命令アドレスステージA0及びB1に、命令供給部IF0及びIF1の命令供給系動作、並びに、命令マルチプレクサMX0及びMX1の動作が命令フェッチステージI0及びI1に、命令デコーダDEC0及びDEC1の動作が命令デコードステージD0及びD1に、命令実行部EX0及びEX1の動作が命令実行ステージE0及びE1に、メモリ制御部MCの動作がロードステージL1、L2、及び、L3に対応する。レジスタスコアボードRSは命令デコード、実行、ロードの各ステージの情報を保持し更新している。レジスタモジュールRMは命令デコードステージD0及びD1での読出しデータ供給と、命令実行ステージE0及びE1、並びに、ロードステージL3の後のデータ書き戻し時に動作

する。

図20は図19のプロセッサの命令供給部IFj（j=0,1）の例である。通常動作時は、+4インクリメンタによってプログラムカウンタPCjから次プログラムカウンタPCj+4を生成し、マルチプレクサMXj及びMRjによって命令アドレスIAjとして選択して出力し、プログラムカウンタPCjにも格納する。この処理を繰り返すことにより、命令アドレスIAjを4づつインクリメントしていき、連続アドレスの命令のフェッチを要求する。命令アドレスIAjからフェッチされた命令ILは命令キューIQjn（nはエントリ番号）に格納する。命令を格納する時は常にプログラムカウンタPCjnにPCjを、後述するリピート回数RCjを格納し、有効ビットIVjnをアサートする。

分岐命令デコーダBDECjは命令キューIQjnから分岐系命令（分岐,THRDG,THRDE,LDRS，LDRE,LDRC等）を取り出してデコードし、オフセットOFSj及びスレッド生成信号GTH0又はスレッド終了ETH1を出力する。そして、加算器ADjにおいてプログラムカウンタPCjnとオフセットOFSjを加算する。

命令が分岐命令又はスレッド生成命令THRDGの場合は、命令アドレスマルチプレクサMXj及びMRjは加算器ADj出力を分岐先アドレスとして選択し、命令アドレスIAjへ出力し、プログラムカウンタPCjにも格納する。そして、命令アドレスIAjからフェッチされた命令ILを分岐命令ならば命令キューIQjnに、スレッド生成命令THRDGならばIF1の命令キューIQ1nに格納する。命令供給部IF0は命令がスレッド生成命令THRDGの場合は更にスレッド生成GTH0をアサートし、命令供給部IF1を起動する。命令供給部IF1は命令がスレッド終了命令ETHRDの場合にスレッド終了ETH1をアサートして、停止する。

命令が図2のLDRS命令の場合は加算器ADj出力をリピート開始アドレスRSjに格納する。命令が図2のLDRE命令の場合は加算器ADj出力をリピート終了アドレスREjに格納する。命令が図2のLDRC命令の場合は、オフセットOFSjをリピート回数としてリピート回数マルチプレクサMCjで選択し、リピート回数RCjに格納する。リピート回数は1回以上とし、0回を指定しても1回実行してから抜けるものとする。同時に、リピート命令数用比較器CRjによって、リピート開始アドレスRSjとリピート終了アドレスREjを比較し、一致した場合は1命令リピートであるから、そ

の1命令を命令キューIQjnに保持し続けて、命令フェッチを抑止する。
リピート機構不使用時はリピート回数RCjを0にしておく。この時、回数比較器CCjにはリピート回数RCjの最下位ビット以外が入力されて0と比較される。比較結果は一致なので、リピート終了検出用比較器CEj出力がANDゲートによってマスクされ、命令アドレスマルチプレクサMRjは、リピート終了検出用比較器CEjへの入力PCj及びREjの値に依らずに命令アドレスマルチプレクサMXj出力を選択し、リピート処理は行われない。
リピート開始アドレスRSj及びリピート終了アドレスREjにアドレスを格納し、リピート回数RCjに2以上の値を格納するとリピート機構が動作する。リピート終了検出用比較器CEjにおいて常時プログラムカウンタPCjとリピート終了アドレスREjが比較され、一致信号がANDゲートに出力される。プログラムカウンタPCjとリピート終了アドレスREjが一致すると一致信号が1となる。この時、リピート回数RCjが2以上であれば、リピート終了検出用比較器CEj出力が0となるので、ANDゲート出力が1となり、命令アドレスマルチプレクサMRjは、リピート開始アドレスRSjを選択し、命令アドレスIAjとして出力する。この結果、命令フェッチはリピート開始アドレスに戻る。上記動作と同時に、リピート回数RCjがデクリメントされ、その結果がリピート回数マルチプレクサMCjで選択され、リピート回数RCjの入力となる。リピート回数RCjはプログラムカウンタPCjとリピート終了アドレスREjが一致し、かつリピート回数RCjが0でなければ更新する。命令キューIQjnではキュー内の各命令に対応するリピート回数RCjをスレッド同期番号IDjnとして付与する。リピート回数RCjが1になったら、回数比較器CCj出力が1となってリピート処理が行われなくなり、リピート回数RCjが0に更新されて終了する。1命令リピートの場合は、命令を命令キューIQjnに保持し続け、スレッド同期番号IDjnのみ更新する。そして、リピート終了時に通常の命令キューIQnj動作に戻る。
尚、スレッド同期番号IDjnはリピート回数RCjの下位数ビットにすることも可能である。この場合、データ定義スレッドが先行し過ぎるとリピート回数RC0とRC1とが異なるにもかかわらずスレッド同期番号ID0nとID1m（mはエントリ番号）が一致する可能性がある。ような場合はデータ定義スレッドの命令フェッチを抑止する。即ち、スレッド同期番号ID0nとID1mが一致し、かつ、リピート回数RC0とR

C1とが異なる場合、IF0は命令フェッチしない。

図21は図19のプロセッサの命令マルチプレクサMj（j = 0,1）の例である。命令Ix（x = j0,k1,j）はオペコードOPx、レジスタフィールドRAx、RBx、スレッド同期番号IDx、及び命令有効IVxから成る。命令マルチプレクサMjは２つの命令Ij0及びIk1（{j,k} = {0,1},{1,0}）から命令Ij0が実行可能ならば命令Ij0を、そうでなければ命令Ik1を命令Ijとして選択する。そして、選択したスレッドをスレッド番号THjとして出力する。即ち、命令Ij0を選択すればTHj = j、命令Ik1を選択すればTHj = kとする。命令Ijのうち、オペコードOPj及び命令有効IVjは命令コードMIjとして命令デコーダDECjへ、レジスタフィールドRAj、RBj、スレッド同期番号IDj、及びスレッド番号THjはレジスタ情報MRjとしてレジスタスコアボードRSへ出力する。

実行可能性は先行実行されている命令とのデータ依存関係によって判断する。図17のようにロードレイテンシ4のパイプライン構成では、先行３命令とのフロー依存によって実行不能になる可能性がある。図21のTHj生成論理がこのフロー依存判定と命令の有効性判定を行っている。本論理は後述するレジスタスコアボードRSと同様な論理である。レジスタスコアボードRSからスコアボード情報CMを受け取って判定を行う。まず、命令コードOPj0がレジスタフィールドRAj0及びRBj0をレジスタ読出しに使用するかをチェックし、読出し有効MVAj及びMVBjを生成する。readRA及びreadRBはこのための関数であり、命令のコード割付が規則的ならば命令コードOPj0の一部をチェックするだけで高速に判定可能である。また、式を統一するために、書き戻し可能Ry（y = L,L0,L1）のうち、本来は存在しないRLを、RL=0と定義する。この時、フロー依存検出MFjyは図21のようになる。フロー依存は、同一スレッド、同一スレッド同期番号、又はレジスタファイルへの書き戻し可能な場合に、有効な読出しと書込みのレジスタ番号が一致したら発生する。そして、フロー依存が発生せず命令が有効であれば、選択有効MVjをアサートし、該MVjに基づいてIj及びTHjを選択する。更に、THj生成論理はデータ使用スレッドがデータ定義スレッドを追い越さないことを保証する。これは、スレッド同期番号IDj0とIDk1とが一致した場合にTHj=0とすることにより実現する。即ち、スレッド同期番号が一致した場合はデータ定義スレッドを選択する。尚、デ

ータ依存関係判定には時間がかかるため、メモリ制御部MCからのフェッチ命令を命令キューIQjnにラッチせずに命令マルチプレクサMjに直接供給した場合は、データ依存関係判定は行わず、実行可能であると予測して供給する。通常、直接供給する場合は分岐先の先頭命令であり実行できる可能性が高い。

上記選択方式により、命令I00及びI10の実行可能性に応じて図22のように命令が選択される。#1の場合、命令I00及びI10が選択され双方とも実行可能である。#2の場合、命令I10が実行不能なので命令I11も実行不能である。一方、選択された命令I00及びI01のうち、I00は実行可能でありI01の実行可能性は不明である。即ち、実行可能な命令又は実行可能性のある命令が選択され、実行不能な命令は選択されない。#3の場合も同様である。#4の場合、命令I00及びI10が実行不能なので、4命令全てが実行不能でありどの命令を選択しても実行しない。

図23はレジスタスコアボードRSの例である。従来プロセッサ同様、パイプラインステージに対応するレジスタファイルへの書込み情報を保持して新規読出し情報と比較し、レジスタに関するフロー依存、逆依存、出力依存の3種類の依存関係を検出する。また、逆依存又は出力依存によって一時的に抑止されているレジスタファイルへの書込み情報を保持して新規読出し情報と比較し、上記3種類の依存関係を検出する。また、逆依存又は出力依存による書込みの可否を判定して書込み指示を出す。詳細は下記の通りである。

スコアボード先頭セルSBL0は、マルチプレクサMLがレジスタ情報MR0又はMR1から選択したロードデータ書込み情報RLをロードステージL0の制御情報として保持し、該保持データとレジスタ情報MR0及びMR1とからバイパス制御情報BPL0y（y = RA0,RB0,RA1， RB1）及び次ステージ制御情報NL0を生成して出力する。同様に、スコアボード先頭セルSBE0及びSBE1は、それぞれレジスタ情報MR0及びMR1を演算ステージE0及びE1の制御情報として保持し、該保持データとレジスタ情報MR0及びMR1とからバイパス制御情報BPE0y及びBPE1y、並びに次ステージ制御情報NE0及びNE1を生成して出力する。また、スコアボード非先頭セルSBL1,SBL2,及びSBL3は、それぞれ次ステージ制御情報NL0,NL1,及びNL2をロードステージL1,L2,及びL3の制御情報として保持し、該保持データとレジスタ情報MR0及びMR1とからバイパス制御情報BPL1y,BPL2y,及びBPL3y、並びに次ステージ制御情報NL1,NL2,及びN

L3を生成して出力する。更に、スコアボード非先頭セルSBTB0,SBTB1,及びSBTB2は、それぞれスコアボード制御部CTLによって選択された一時バッファ制御情報NM0,NM1,及びNM2を一時バッファ制御情報として保持し、該保持データとレジスタ情報MR0及びMR1とからバイパス制御情報BPTB0y,BPTB1y,及びBPTB2y、並びに次サイクル制御情報NTB0,NTB1,及びNTB2を生成して出力する。また、スコアボード制御部CTLはフロー依存及び一時バッファフルによるストール検出、レジスタファイルRF及び一時バッファTBへの書込み制御を行う。また、スコアボードセルSBL0,SBL1,及びSBL2への入力信号を、スコアボード情報CM = {RL,THL,IDL,VL,NL0,NL1} として命令マルチプレクサMX0及びMX1へ出力する。

以下、マルチプレクサML、スコアボード先頭セルSBL0,SBE0,及びSBE1、スコアボード非先頭セルSBL1,SBL2,SBL3,SBTB0,SBTB1,及びSBTB2、並びに、スコアボード制御部CTLの詳細を図24から図27によって説明する。

図24はマルチプレクサMLの例である。レジスタ情報MR0又はMR1からロード命令の書込み情報を選択する。双方ともロード命令の場合は先行命令の情報を選択する。双方ともロード命令でない場合はどちらを選択しても良い。したがって、先行命令がロード命令ならばそのレジスタ情報を、ロード命令でなければもう一方のレジスタ情報を選択する。前述のようにレジスタ情報MRj（j = 0,1）はレジスタフィールドRAj、RBj、スレッド同期番号IDj、及びスレッド番号THjから成る。後述のようにスレッド番号TH0が0ならば命令I0が、1ならば命令I1が先行命令である。図24のレジスタ情報MR0の選択条件式の第1項は、TH0==0かつ書込み信号LV0アサートなので、命令I0が先行命令かつロード命令である。一方、第2項はTH0==1かつ書込み信号LV1ネゲートなので、命令I1が先行命令かつ非ロード命令である。どちらを選択したかを示すロードパイプSBLをスコアボード制御部CTLに出力する。マルチプレクサMLの説明時に説明したようにスレッド番号TH0が0ならば命令I0が、1ならば命令I1が先行命令である。また、ストール時には命令が実行されないので書込み有効VLをストールSTL0又はSTL1で無効化する。

スレッド番号TH0が0ならば、命令マルチプレクサMX0の選択命令組合せは図22の#1又は#2である。#1ならば命令I0は命令供給部IF0から供給されたデータ定義スレッドの命令I00であり、命令I1は命令供給部IF1から供給されたデータ使用スレッ

ドの命令I10である。したがって、命令I00を命令I10より先に実行すれば、本発明のデータ定義スレッドとデータ使用スレッドとの実行順序ルールに違反しない。#2ならば命令I0とI1は命令I00とI01であり、逐次実行順序はI0の方が先である。一方、スレッド番号TH0が1ならば、命令マルチプレクサMX0の選択命令組合せは図22の#3又は#4である。#3ならば命令I0とI1は命令I11とI10であり、逐次実行順序はI1の方が先である。#4は命令I0とI1の双方とも実行不能である。以上より、スレッド番号TH0が0ならば命令I0が、1ならば命令I1が先行命令である。

図25はスコアボード先頭セルSBx（x = L0,E0,E1）の例である。入力Rs,THt,IDt,及びVt&¯u（{s,t,u} = {L,L,1},{A0,0,STL0},{A1,1,STL1}）をxステージ書込み情報である書込みレジスタ番号Wx、書込みスレッド番号THx、書込みスレッド同期番号IDx、及び書込み有効Vxとして保持し、これらと、レジスタ情報MR0及びMR1、レジスタ書込み信号V0及びL0、並びに、V1及びL1とから、バイパス制御情報BPxy（y = RA0,RB0,RA1,RB1）及び次ステージ書込み制御情報Nx = {Wx,THx,IDx,BNx,Vx} を生成して出力する。入力Vtをuでマスクするのはストール時には命令が実行されないので書込み情報を無効化するためである。

図25の論理部SBxLの第1式がバイパス制御情報BPxyの定義式である。バイパス制御情報BPxyは、xステージの書込みが有効であり、かつ、書込みレジスタ番号Wxとレジスタ読出し番号yとが一致し、かつ、書込みと読出しが同一スレッド番号又は同一スレッド同期番号である場合にアサートする。同一スレッド番号の場合はスレッド内のバイパス制御であり従来のプロセッサにおいても一般的に行われている。一方、同一スレッド同期番号の場合は、データ定義スレッドからデータ使用スレッドへのバイパス制御である。逆方向のデータ使用スレッドからデータ定義スレッドへのバイパス制御が発生しないのは、データ使用スレッドがデータ定義スレッドを追い越さないように、命令マルチプレクサMjが構成されているからである。

次ステージ書込み制御情報Nxのうち、書込みレジスタ番号Wx、書込みスレッド番号THx、書込みスレッド同期番号IDx、及び書込み有効Vxは保持した情報をそのまま出力する。書き戻しBNxは逆依存及び出力依存が解消されレジスタファイルへの書き戻しが可能であることを示す。本実施例ではデータ使用スレッドのスレッ

ド同期番号が書込み制御情報のスレッド同期番号と一致したらアサートし、書き戻すまでアサートし続ける。図25の論理部SBxLの第2式が書き戻しBNxの定義式である。

図26はスコアボード非先頭セルSBx（x = L1,L2,L3,TB0,TB1,TB2）の例である。入力信号Wt,THt,IDt,BNt及びVt（t = L0,L1,L2,M0,M1,M2）をxステージ書込み情報である書込みレジスタ番号Wx、書込みスレッド番号THx、書込みスレッド同期番号IDx、書き戻しBx、及び書込み有効Vxとして保持し、これらと、レジスタ情報MR0及びMR1、レジスタ書込み信号V0及びL0、並びに、V1及びL1とから、バイパス制御情報BPxy（y = RA0,RB0,RA1, RB1）及び次ステージ書込み制御情報Nx = {Wx,THx,IDx,BNx,Vx} を生成して出力する。

図26の論理部SBxLの第1式がバイパス制御情報BPxyの定義式である。バイパス制御情報BPxyは、xステージの書込みが有効であり、かつ、書込みレジスタ番号Wxとレジスタ読出し番号yとが一致し、かつ、書込みと読出しが同一スレッド番号、同一スレッド同期番号、又は書き戻しアサート中である場合にアサートする。図25との違いは書き戻しBxアサート中という条件が加わっていることである。この条件によって、書き戻しの完了していないデータをレジスタ値の代わりにバイパスして供給する。図26の論理部SBxLの第2式が書き戻しBNxの定義式である。図25との違いは書き戻しBxアサート中という条件が加わっていることである。この条件によって、書き戻しBxは一度アサートされると書き戻されるまでアサートされ続ける。

図27は図23のスコアボード制御論理CTLの例である。フロー依存によるストールは以下のように検出する。ロードレイテンシが4であることにより、書込み制御情報NLz（z = 0,1,2)に対応するデータはまだ有効ではない。したがって、バイパス制御BPzy（y = A0,A1,B0,B1)がアサートされると有効でないデータのバイパスが必要であり、実現不可能である。このため、これらの信号がアサートされた場合はデータが有効になるまでバイパスデータを使用する命令の実行開始を待たせる必要がある。このためにバイパス制御BPzyを集めたストールSTL0及びSTL1を出力する。この際、レジスタ情報有効VR0及びVR1のうち読出し有効VA0,VB0,VA1及びVB1でバイパス制御BPzyをマスクする。更に、先行命令がストールすると逐

次実行順序維持のために後行命令もストールする。マルチプレクサMLの説明時に説明したようにスレッド番号TH0が0ならば命令I0が、1ならば命令I1が先行命令である。また、先行、後行双方の命令がデータロード命令の場合、後攻命令をストールする。マルチプレクサMLで選ばれなかったパイプ、即ちロードパイプSBLの指さないパイプかつデータロード用書込みレジスタRB0又はRB1への書込み有効LV0又はLV1がアサートされた場合はストールする。以上より、ストール信号STL0及びSTL1は図27の第1式から第4式によって定義される。単独スレッドSTHはスレッド生成GTH0からスレッド終了ETH1の間ネゲートされる。したがって、生成式は図27の第5式のようになる。

書込みデータはパイプラインステージE0,E1,又はL3終了時に有効となる。これに対応するレジスタスコアボードRSの書込み情報はNE0,NE1,又はNL3である。また、一時バッファに保持されているデータも有効である。有効なデータは逆依存又は出力依存が解消し次第、レジスタファイルRFに書き戻す。スレッド番号THx（x = E0,E1,L3,TB0,TB1,TB2）が1の場合はデータ使用スレッドなので、逆依存又は出力依存は発生せず、有効なデータは常に書込んでよい。一方、スレッド番号THxが0の場合は、逆依存又は出力依存が解消して書き戻しBxがアサートされたら書き戻す。更に、単独スレッドSTHアサート中は逆依存又は出力依存は発生しない。以上より、書込み指示Sxは図27の第6式のようになる。有効なデータが逆依存又は出力依存によって書込み不能な場合は、一時バッファ制御Cxをアサートし、一時バッファTBへの書込みを行う。一時バッファ制御Cxは図27の第7式のようになる。一時バッファTBは3エントリであるため、6本の一時バッファ制御Cxのうち4本以上アサートされた場合は一時バッファTBへの書込みが不可能である。この場合は一時バッファ起因のストール信号STLTBをアサートし、パイプラインの進行を止める。3本以下の場合は書込みが可能である。一時バッファTBへの書込みはデータ定義スレッドからのみ行われるので、書込まれるデータには逐次実行順序がある。この順序が常に、早い方からTB2,TB1,TB0となり、かつ、一時バッファTBの1エントリを使用する場合はTB0を、2エントリ使用する場合はTB0,TB1を使用するように、一時バッファTBへの書込みデータを選択する。この方針によってデータ選択M0,M1,M2を生成すると図27の表のようになる。尚、パイプラインステ

ージE0,E1,又はL3からの書込みデータを含めた逐次実行順序は早い方からTB2,TB1,TB0,L3,E0,E1である。そして、データ選択M0,M1,M2によって次ステージ書込み制御情報Nt（t = M0,M1,M2）をNxから選択する。図27の最後の3式が選択式である。図28は図19のプロセッサのレジスタモジュールRMの例である。レジスタファイルRF、一時バッファTB、及び読出しデータマルチプレクサMy（y = A0,A1,B0,B1)から成る。レジスタ制御信号CR及び出力データDE0,DE1,及びDL3を入力とし、読出しデータDRy（y = A0,A1,B0,B1)を出力とする。レジスタ制御信号CRはレジスタ読出し番号Ry、バイパス制御BPxy（x = E0,E1,L3,TB0,TB1,TB2)、レジスタ書込み番号Wx、レジスタ書込み制御信号Sx、一時バッファ書込みデータ選択Mz（z = 0,1,2)、及びスレッド番号TH0から成る。

レジスタファイルRFは16エントリ、4読出し、6書込みである。書込み制御信号Sxがアサートされるとデータ Dxをレジスタファイル RFのWx番に書込む。また、レジスタファイルRFのRy番をレジスタ読出しデータRDyとして読出す。

一時バッファTBはバイパス制御BPTBzy、データ選択Mz、並びに出力データDE0,DE1,及びDL3を入力とし、一時バッファ保持データDTBz及び一時バッファ読出しデータTByを出力する。また、書込みデータ選択Mzに従って保持データDTBzを更新する。詳細は図29を用いて説明する。一時バッファ保持データDTBzは常に出力している。書込みデータDNTBzの選択論理は一時バッファマルチプレクサTBMの最初の3式である。選択信号Mzに従って選択する。読出しデータTByの選択論理は一時バッファマルチプレクサTBMの最後の式である。バイパス制御BPTBzyに従って選択する。尚、複数のバイパス制御BPzyがアサートされた場合、最も新しいデータを選択する。即ち、逐次実行順序の最も遅いものを選択する。

読出しデータマルチプレクサMyはバイパス制御BPxy、スレッド番号TH0、レジスタ読出しデータRDy、一時バッファ読出しデータTBy、並びに、出力データDE0,DE1,及びDL3を入力とし、読出しデータDRy（y = A0,A1,B0,B1)を出力する。詳細は図30を用いて説明する。複数のバイパス制御BPxyがアサートされた場合最も新しいデータを選択する。出力データDE0とDE1はスレッド番号TH0が0ならばDE1が新しく、1ならばDE0が新しい。この結果、選択論理は図30の左側の囲いの中の論理となる。この時、一時バッファバイパス制御BPTByは図30の右側の囲いの中の論

理のように3個のバイパス制御BPTBzyの論理和となる。

さて、本実施例によって実際に図16のプログラムを実行すると以下のような動作となる。まず、時刻t0では、命令#1及び#2の命令アドレスステージA0を行う。命令供給部IF0が命令#1のアドレスを命令アドレスIA0に載せ、メモリ制御部MCにフェッチ要求を出す。同時に命令アドレスIA0をプログラムカウンタPC0にラッチする。そして、命令アドレスマルチプレクサMIAがIAとしてIA0を選択し、メモリ制御部MCへ出力する。

次サイクル時刻t1では、命令#3及び#4の命令アドレスステージA0を行う。プログラムカウンタPC0に4を加えて命令アドレスIA0に載せマルチプレクサMIA経由でメモリ制御部MCへ出力し、フェッチ要求を出す。同時に命令アドレスIA0をプログラムカウンタPC0にラッチする。更に、命令#1及び#2の命令フェッチステージI0を行う。メモリ供給部MCは命令#1のアドレスから2命令、即ち命令#1及び#2をフェッチし、フェッチ命令ILとして命令供給部IF0へ出力する。命令供給部IF0はこれを命令キューIQ0nに格納すると同時に命令I00及びI01として命令マルチプレクサMX0及びMX1に供給する。この時、リピートカウンタRC0はリピート機構不使用時の0であるからスレッド同期番号ID00及びID01として0を付与する。命令マルチプレクサMX0及びMX1はそれぞれ命令I00及びI01を選択して、命令コードMI0及びMI1、並びに、レジスタ情報MR0及びMR1を生成し、命令デコーダDEC0及びDEC1、並びに、レジスタスコアボードRSへ出力する。即ち、命令#1及び#2はそれぞれパイプ0及びパイプ1に供給される。尚、命令#1は分岐系命令であるが、命令フェッチ直後に供給する場合は分岐系命令デコーダBDEC0の解析前に供給するため、命令デコーダDEC0に供給し、命令デコーダDEC0で処理をノーオペレーション（NOP）化する。

時刻t2では、命令#5、#6及び#9の命令アドレスステージA0を行う。まず、命令供給部IF0のプログラムカウンタPC0に4を加えて更新し、命令#5及び#6のフェッチ要求を出す。また、命令#9はリピート開始かつ終了命令であるため、命令#1、#3、及び#5によってリピートセットアップを行う。分岐系命令デコーダBDEC0が命令#1のLDRE命令をデコードし、プログラムカウンタPC0と命令#9へのオフセットOFS0を加算して命令#9のアドレスを生成し、リピート終了アドレスRE0に格納する

。また、時刻t1と同様に命令#3及び#4の命令フェッチステージI0を行う。更に、命令#1及び#2の命令デコードステージD0及びD1動作として以下を行う。命令デコーダDEC0は命令#1が分岐系命令であるため処理をNOP化する。命令デコーダDEC1は命令#2をデコードして、制御情報C1を出力し、レジスタ情報有効VR1を出力する。命令#2は定数x_addrをr0に格納する命令である。通常、アドレスは32ビットあるが、ここではx_addr及び後述するy_addrは命令内の即値で表現できる小さなアドレスであるものとする。そこで、即値x_addrを制御情報C1に載せて命令実行部EX1に供給する。また、RA1をr0への書込み制御に使用するのでレジスタ情報有効VR1のうちV1をアサートする。レジスタスコアボードRSでは、スコアボードセルSBE1に命令#2の書込み情報を格納する。

時刻t3では、命令#7、#8及び#9の命令アドレスステージA0動作として以下を行う。まず、時刻t2同様、命令#7及び#8のフェッチ要求を出す。分岐系命令デコーダBDEC0では命令#3のLDRS命令をデコードし、プログラムカウンタPC0と命令#9へのオフセットOFS0を加算して命令#9のアドレスを生成し、リピート開始アドレスRS0に格納する。同時に、リピート開始アドレスRS0とリピート終了アドレスRE0をリピートアドレス比較器CR0で比較し、どちらも命令#9を指していて一致し、1命令リピートであるので、この一致情報を記憶する。また、時刻t1と同様に命令#5及び#6の命令フェッチステージI0を行う。更に、命令#3及び#4の命令デコードステージD0及びD1動作として以下を行う。命令デコーダDEC0は命令#3が分岐系命令であるため処理をNOP化する。命令デコーダDEC1は命令#4が定数y_addrをr1に格納する命令なので、即値y_addrを制御情報C1に載せて命令実行部EX1に供給する。また、RA1をr1への書込み制御に使用するのでレジスタ情報有効VR1のうちV1をアサートする。また、命令#2の命令実行ステージE1を行う。命令実行部EX1は制御情報C1に従って命令#2を実行する。即ち、即値x_addrを実行結果DE1に出力する。

レジスタスコアボードRSは、スコアボードセルSBE1から命令#2の書込み情報を出力し、制御部CTLにおいて単独スレッドSTHかつ書込み有効VE1であるから、レジスタ書込み信号SE1をアサートする。この結果、レジスタモジュールRMのレジスタファイルRFに実行結果DE1である即値x_addrが書込みレジスタ番号WE1によって

指定されるr0に書込まれる。また、スコアボードセルSBE1に命令#4の書込み情報を格納する。

時刻t4では、命令#11及び#12の命令アドレスステージA0及びA1動作として以下を行う。命令供給部IF0の分岐系命令デコーダBDEC0では命令#5のTHRDG/R命令をデコードし、PC0に命令#11へのオフセットOFS0を加えて新スレッドの先頭アドレス、即ち、命令#11のアドレスを生成して命令アドレスIA0に載せ、メモリ制御部MCに命令フェッチ要求を出す。また、時刻t1と同様に命令#7及び#8命令フェッチステージI0を行う。また、命令デコードステージD0及びD1動作として以下を行う。命令デコーダDEC0は命令#5が分岐系命令であるため処理をNOP化する。命令デコーダDEC1は命令#6をデコードして、命令#2同様、即値0を制御情報C1に載せて命令実行部EX1に供給し、レジスタ情報有効VR1のうちV1をアサートする。また、時刻t3の命令#2と同様に命令#4の命令実行ステージE1を行う。レジスタスコアボードRS及びレジスタモジュールRMでは、時刻t3の命令#2及び#4同様、命令#4及び#6の処理を行う。

時刻t5では、命令#9及び#10の命令アドレスステージA0動作として以下を行う。まず、時刻t2同様、命令#9及び#10のフェッチ要求を出す。命令供給部IF0の分岐系命令デコーダBDEC0では命令#7のLDRC命令をデコードし、OFS0にリピート回数8を載せ、リピート回数RC0に格納する。これでリピートセットアップ完了である。また、命令#11及び#12の命令フェッチステージI1を行う。メモリ制御部MCは命令#11及び#12をフェッチし、命令供給部IF1はこれにスレッド同期番号ID1nとして0を付加して命令キューIQ1nに保持すると共に、命令I10及びI11として命令マルチプレクサMX1及びMX0に供給する。しかし、命令マルチプレクサMX1及びMX0は命令供給部IF0側のデータ定義スレッドと命令供給部IF1側のデータ使用スレッドのスレッド同期番号がどちらも0で一致しているため、図21の選択論理によりデータ定義スレッドである命令供給部IF0側を選択する。この時、命令キューIQ0nは空なので、命令デコーダDEC0及びDEC1には無効な命令が供給される。また、命令#7及び#8の命令デコードステージD0及びD1動作として以下を行う。命令デコーダDEC0は命令#7が分岐系命令であるため処理をNOP化する。命令デコーダDEC1は命令#8をデコードしNOP制御を出力する。更に、時刻t3の命令#2と同様に命令#6

の命令実行ステージE1を行う。レジスタスコアボードRS及びレジスタモジュールRMでは、時刻t3の#4同様、命令#6の処理を行う。

時刻t6では、命令#9の命令アドレスステージA0を行う。命令供給部IF0において、プログラムカウンタPC0とリピート終了アドレスRE0が一致し比較器CE0の出力が1となり、リピート回数RC0は8なので比較器CC0出力は0となり、AND出力が1となるので、マルチプレクサMR0がリピート開始アドレスRS0を選択し、これを命令フェッチアドレスIA0として出力してプログラムカウンタPC0に格納する。また、リピート回数RC0をデクリメントして7とし、マルチプレクサMC0で選択してリピート回数RC0に格納する。更に、1命令リピートなので命令キューIQ0nに命令#9以降の保持を指示する。また、命令#13、#14及び#15の命令アドレスステージA1を行う。命令供給部IF1のプログラムカウンタPC1に4を加えて更新し、命令#13及び#14のフェッチ要求を出す。分岐系命令デコーダBDEC1では命令#11のLDRE命令をデコードし、命令#5と同様に命令#15のアドレスをリピート終了アドレスRE1に格納する。また、時刻t1と同様に命令#9及び#10の命令フェッチステージI0を行う。この時、スレッド同期番号ID0として0を付加する。尚、リピート動作の初回はリピート終了アドレスRE0に達した時に判明するので、スレッド同期番号は8ではなくリピート範囲到達前と同じ0とする。また、命令保持指示が出ているので供給後も命令キューIQ0nに命令#9及び#10を保持する。尚、命令#11及び#12が命令キューIQ1nに保持されていて、分岐系命令デコーダBDEC1が命令#11及び#12を解析して、共に分岐系命令であることを判断する時間があり、他に命令がないので、命令キューIQ1nには命令デコーダに供給する命令がなく、命令フェッチステージI1で処理される命令はない。

時刻t7では、命令#9及び#15の命令アドレスステージA0及びA1を行う。命令供給部IF0は前サイクルと同様にリピート動作しリピート回数RC0を6にする。命令供給部IF1の分岐系命令デコーダBDEC1では命令#12のLDRS命令をデコードし、命令#3と同様に命令#15のアドレスをリピート開始アドレスRS1に格納し、1命令リピート制御のためのアドレス一致情報を記憶する。また、命令#9、#13及び#14の命令フェッチステージI0及びI1を行う。命令供給部IF0は命令キューIQ0nに保持している命令#9にスレッド同期番号ID00として7を付けて、命令I00として命令マルチ

プレクサMX0に供給する。尚、本動作は上記デクリメントと同時にデクリメント前の値を使用して行われる。このため付加される値は7である。リピート動作により命令#9の次命令は命令#10ではないので、命令I01として供給する命令はなく、命令I01の命令有効IV01はネゲートする。メモリ制御部MCは命令#13及び#14をフェッチし、命令供給部IF1はこれにスレッド同期番号ID1nとして0を付加して命令キューIQ1nに格納すると同時に、命令I10及びI11として命令マルチプレクサMX1及びMX0に供給する。この時、命令I00として供給される命令#9はレジスタ読出しを伴うが、先行するデータロード命令がないため、スコアボード情報CMの書込み有効VL，VL0,及びVL1は全てネゲートされており、フロー依存は発生しない。また、命令#13はフェッチ直後であるため実行可能性判定をしない。この結果、命令マルチプレクサMX1及びMX0は命令I00及びI10、即ち命令#9及び#13を選択し命令デコーダDEC0及びDEC1に供給する。また、命令#9の命令デコードステージD0を行う。命令デコーダDEC0は命令#9がレジスタr0の指すアドレスからデータロードしてレジスタr2に格納し、レジスタr0をインクリメントする命令なので、その制御情報C0を出力する。また、RA0をr0の読出し及び書込み制御に、RB0をr2への書込み制御に使用するのでレジスタ情報有効VR1のうちVA0、V0、及びLV0をアサートする。

レジスタスコアボードRSは、レジスタ読出し番号RA0及び、バイパス制御BPxy（x = E0,E1， L0,L1,L2,L3,TB0,TB1,TB2，y = A0,B0,A1,B1）を出力する。図18のパイプライン動作の下に各時刻における各スコアボードセルの書込み及び読出しのレジスタ番号及びスレッド同期番号を付加した。ハッチングをかけた部分がスレッド1（データ使用スレッド）、他がスレッド0（データ定義スレッド）の情報である。時刻t7では有効な書込み情報がないのでバイパス制御BPxyは全てネゲートされる。また、スコアボードセルSBE0及びSBL0に命令#9のr0及びr2への書込み情報を格納する。スコアボードセルSBL0入力の選択は、図24のような論理であり、スレッド番号TH0==0かつレジスタ情報有効LV0がアサートなので、パイプ0側の命令#9の情報が選択される。

時刻t8では、命令#9、#15及び#16の命令アドレスステージA0及びA1を行う。命令供給部IF0は前サイクルと同様にリピート動作しリピート回数RC0を5にする。命

令供給部IF1のプログラムカウンタPC1に4を加えて更新し、命令#15及び#16のフェッチ要求を出す。分岐系命令デコーダBDEC1では命令#13のLDRC命令をデコードし、命令#7と同様にリピート回数RC1に8を格納する。また、命令#9及び#14の命令フェッチステージI0及びI1を行う。命令供給部IF0は時刻t7同様、命令#9にスレッド同期番号ID00として6を付けて、命令I00として命令マルチプレクサMX0に供給する。この時、命令#9はレジスタr0の読出しを伴いフロー依存発生の可能性がある。しかし、スコアボード情報CMの書込み有効VLがアサートされている先行データロードはr2に対するものなので、レジスタ番号不一致によりフロー依存は発生しない。また、命令供給部IF1は命令キューIQ1nに保持している命令#14を命令I00として命令マルチプレクサMX0に供給する。この結果、命令マルチプレクサMX0及びMX1は命令I00及びI10、即ち命令#9及び#14を選択し命令デコーダDEC0及びDEC1に供給する。また、時刻t7と同様に命令#9の命令デコードステージD0を行う。また、命令#13のデコードステージD1を行う。命令デコーダDEC1は命令#13が分岐系命令であるため処理をNOP化する。更に、命令#9の命令実行ステージE0を行う。命令実行部EX0は制御情報C0に従って、読出しデータDRA0をロードアドレスとして実行結果DM0に載せてメモリ制御部MCに出力する。また、読出しデータDRA0をインクリメントし実行結果DE0としてレジスタモジュールRMに出力する。

レジスタスコアボードRSでは、図18のように時刻t8ではセルSBE0及びSBL0にそれぞれレジスタr0及びr2への書込みがスレッド同期番号0で記憶されている。また、レジスタ読出し番号RA0にr0がスレッド同期番号7で出力されている。セルSBE0と読出し番号RA0がr0で一致し、スレッド同期番号は0と7で異なるもののスレッド番号THE0とTH0が共に0で一致するので、バイパス制御のうちBPE0A0がアサートされる。また、スコアボードセルSBE0及びSBL0ではスレッド番号THE0及びTHL0が1であるため図25の論理により書き戻しBNE0及びBNL0がネゲートされる。この書き戻しBNL0を付加して生成された次ステージ書込み制御情報NL0はスコアボードセルSBL1に格納される。また、制御論理CTLにおいて、単独スレッドSTHはネゲートされており、スレッド番号THE0が0で上記書き戻しBNE0のネゲートされているので、図27の第6及び第7式により書込み指示SE0はネゲートされ、一時バッファ制御CE0はアサートされる。他のSx（x = TB0,TB1,TB2,L3,E0,E1）及びCxは書込

み有効Vxがネゲートされているので全てネゲートされる。この結果、図27の表により、データ選択M0,M1及びM2はそれぞれE0,TB0及びTB1となる。そして、次ステージ書込み制御情報NM0,NM1及びNM2はそれぞれNE0,NTB0及びNTB1となり、これらが一時バッファ制御情報SBTB0,SBTB1及びSBTB2に格納される。更に、セルSBE0及びSBL0には時刻t7と同様に命令#9の書込み情報を格納する。レジスタモジュールRMでは、データ選択M0,M1及びM2に従って、一時バッファDTB0，DTB1及びDTB2に実行結果DE0，一時バッファデータDTB0及びDTB1が書込まれる。また、バイパス制御BPE0A0がアサートされたので、バイパスマルチプレクサMA0において、図30の論理により実行結果DE0が読出しデータDRA0として選択される。

時刻t9では、命令#9及び#15の命令アドレスステージA0及びA1を行う。命令供給部IF0は前サイクルと同様にリピート動作しリピート回数RC0を4にする。命令供給部IF1ではプログラムカウンタPC1とリピート終了アドレスRE1が命令#15のアドレスで一致し、命令#9同様リピート動作を開始し、リピート回数RC0を7にする。また、命令#9、#15及び#16の命令フェッチステージI0及びI1を行う。命令供給部IF0は時刻t7同様、命令#9にスレッド同期番号ID00として5を付けて、命令I00として命令マルチプレクサMX0に供給する。この時、命令#9はレジスタr0の読出しを伴うが、書込み有効VL及びVL0がアサートされている先行データロードはr2に対するものなので、レジスタ番号不一致によりフロー依存は発生しない。メモリ制御部MCは命令#15及び#16をフェッチし、命令供給部IF1はこれを命令キューIQ1nに格納すると同時に、命令I10及びI11として命令マルチプレクサMX1及びMX0に供給する。命令I10及びI11はフェッチ直後なので命令マルチプレクサMX1は実行可能性判定を行わない。この結果、命令マルチプレクサMX1及びMX0は命令I00及びI10、即ち命令#9及び#15を選択し命令デコーダDEC0及びDEC1に供給する。更に、時刻t7と同様に命令でコーダDEC0は命令#9の命令デコードステージD0を行う。また、命令デコーダDEC1は命令#14の命令デコードステージD1を行う。命令#14はNOPなので制御情報C1はNOP処理となる。更に、時刻t8同様命令#9の命令実行ステージE0を行う。また、メモリ制御部MCで命令#9のデータロードステージL1を行う。

レジスタスコアボードRSは、時刻t9では図18のようになっている。そして、時刻

t8同様バイパス制御BPE0A0がアサートされる。また、セルSBTB0と読出し番号RA0がr0で一致し、スレッド番号THTB0とTH0が共に0で一致するのでバイパス制御BPTB0A0がアサートされる。また、時刻t8同様書き戻しBNE0及びBNL0がネゲートされ、セルSBL1が更新され、書込み指示SE0がネゲートされ、一時バッファ制御CE0がアサートされる。更に、セルSBL1及びSBTB0ではスレッド番号THL1及びTHTB0が1であるため図26の論理により書き戻しBNL1及びBNTB0は引き続きネゲートされる。この書き戻しBNL1を付加して生成された次ステージ書込み制御情報NL1はスコアボードセルSBL2に格納される。そして、図27の第6及び第7式により書込み指示STB0はネゲートされ、一時バッファ制御CTB0はアサートされる。この結果、図27の表により、時刻t8同様データ選択M0,M1及びM2はそれぞれE0,TB1及びTB2となり、これに従って一時バッファ制御情報SBTB0,SBTB1及びSBTB2が更新される。更に、セルSBE0及びSBL0には時刻t7と同様に命令#9の書込み情報を格納する。レジスタモジュールRMでも、時刻t8同様データ選択M0,M1及びM2に従って、一時バッファDTB0,DTB1及びDTB2が更新される。また、バイパス制御BPE0A0及びBPTB0A0がアサートされたので、バイパスマルチプレクサMA0において、図30の論理により実行結果DE0が読出しデータDRA0として選択される。この時、バイパス制御BPTB0A0によって一時バッファTBでは一時バッファ読出しデータTBA0として一時バッファデータDTB0が読出され、バイパスマルチプレクサMA0においてもBPTBA0がアサートされる。しかし、バイパス制御BPE0A0もアサートされているので、図30の論理により新しい実行結果DE0が選択される。

時刻t10では、命令#9及び#15の命令アドレスステージA0及びA1を行う。命令供給部IF0は前サイクルと同様にリピート動作しリピート回数RC0を4にする。命令供給部IF1は前サイクルと同様にリピート動作するが、後述するレジスタスコアボードRSがストールSTL1をアサートするのでリピート回数RC0を7のままとする。また、命令#9、#15及び#17の命令フェッチステージI0及びI1を行う。命令供給部IF0は時刻t7同様、命令#9にスレッド同期番号ID00として4を付けて、命令I00として命令マルチプレクサMX0に供給する。この時、命令#9はレジスタr0の読出しを伴うが、書込み有効VL,VL0及びVL1がアサートされている先行データロードはr2に対するものなので、レジスタ番号不一致によりフロー依存は発生しない。メモ

リ制御部MCは命令#17及びその次の命令をフェッチし、命令供給部IF1はこれを命令キューIQ1nに格納する。また命令#15を命令I10として命令マルチプレクサMX1に供給する。この時、命令I10即ち命令#15はレジスタr2及びr3の読出しを伴うが、書込み有効VL,VL0及びVL1がアサートされているr2に対する先行データロードがスレッド同期番号7,6及び5なのでフロー依存は発生しない。リピート動作により命令#15の次命令は命令#16ではないので、命令I11として供給する命令はなく、命令I11の命令有効IV11はネゲートする。この結果、命令マルチプレクサMX1及びMX0は命令I00及びI10、即ち命令#9及び#15を選択し命令デコーダDEC0及びDEC1に供給する。更に、時刻t7と同様に命令でコーダDEC0は命令#9の命令デコードステージD0を行う。また、命令#15の命令デコードステージD1を行う。命令#15はレジスタr2及びr3を加算してr3に格納する命令なので、その制御情報C1を出力する。また、RA0をr3の読出し及び書込み制御に、RB0をr2の読出し制御に使用するのでレジスタ情報有効VR1のうちVA0、VB0、及びV0をアサートする。また、時刻t8同様命令#9の命令実行ステージE0を行う。更に、メモリ制御部MCで命令#9のデータロードステージL1,L2及びL3を行う。

レジスタスコアボードRSは、時刻t10では図18のようになっている。そして、時刻t9同様バイパス制御BPE0A0及びBPTB0A0がアサートされる。また、セルSBTB1と読出し番号RA0がr0で一致し、スレッド番号THTB1とTH0が共に0で一致するのでバイパス制御BPTB1A0がアサートされる。更に、セルSBL2と命令#15の読出し番号RB1がr2で一致し、スレッド同期番号IDL2とID1が共に0で一致するのでバイパス制御BPL2B1がアサートされる。すると、スコアボード制御部CTLにおいてストールSTL1がアサートされ、命令#15の実行が抑止され、スコアボードセルSBE1へ書込まれる書込み有効がネゲートされる。また、時刻t9同様、書き戻しBNE0,BNL0,BNL1及びBNTB0がネゲートされ、セルSBL1及びSBL2が更新され、書込み指示SE0及びSTB0がネゲートされ、一時バッファ制御CE0及びCTB0がアサートされる。更に、セルSBL2及びSBTB1ではスレッド番号TH1が1かつスレッド同期番号IDL2及びIDTB1とID1とが共に0で一致するため図26の論理により書き戻しBNL2及びBNTB1がアサートされる。この書き戻しBNL2を付加して生成された次ステージ書込み制御情報NL2はスコアボードセルSBL3に格納される。そして、図27の第6及び第7式により書

込み指示STB1はアサートされ、一時バッファ制御CTB1はネゲートされる。この結果、図27の表により、時刻t8同様データ選択M0,M1及びM2はそれぞれE0,TB1及びTB2となり、これに従って一時バッファ制御情報SBTB0,SBTB1及びSBTB2が更新される。更に、セルSBE0及びSBL0には時刻t7と同様に命令#9の書込み情報を格納する。レジスタモジュールRMでも、時刻t8同様データ選択M0,M1及びM2に従って、一時バッファDTB0,DTB1及びDTB2が更新される。そして、書込み指示STB1によりレジスタファイルRFのレジスタr0に一時バッファデータDTB1が書き戻される。また、バイパス制御BPE0A0,BPTB0A0及びBPTB1A0がアサートされたので、バイパスマルチプレクサMA0において、図30の論理により実行結果DE0が読出しデータDRA0として選択される。この時、バイパス制御BPTB0A0及びBPTB1A0によって一時バッファTBでは一時バッファ読出しデータTBA0として一時バッファデータDTB0が読出され、バイパスマルチプレクサMA0においてもBPTBA0がアサートされる。しかし、バイパス制御BPE0A0もアサートされているので、図30の論理により最も新しい実行結果DE0が選択される。

時刻t11では、命令#9及び#15の命令アドレスステージA0及びA1を行う。命令供給部IF0は前サイクルと同様にリピート動作しリピート回数RC0を4にする。命令供給部IF1は時刻t9と同様にリピート動作しリピート回数RC0を6とする。また、命令#9及び#15の命令フェッチステージI0及びI1を行う。命令供給部IF0は時刻t7同様、命令#9にスレッド同期番号ID00として4を付けて、命令I00として命令マルチプレクサMX0に供給する。この時、命令#9には時刻t10同様フロー依存は発生しない。命令供給部IF1は命令#15にスレッド同期番号ID01として7を付けて、命令I10として命令マルチプレクサMX1に供給する。この時、命令#15には時刻t10フロー依存は発生しない。この結果、命令マルチプレクサMX1及びMX0は命令I00及びI10、即ち命令#9及び#15を選択し命令デコーダDEC0及びDEC1に供給する。更に、時刻t7と同様に命令でコーダDEC0は命令#9の命令デコードステージD0を行う。また、命令#15の命令デコードステージD1を行う。命令#15は前サイクルでストールSTL1により実行されなかったので命令でコーダDEC1は入力命令を更新せず、命令#15のデコード結果を再度出力する。また、時刻t8同様命令#9の命令実行ステージE0を行う。更に、メモリ制御部MCで命令#9のデータロードステージL1,L2及びL3を

行う。

レジスタスコアボードRSは、時刻t11では図18のようになっている。尚、前サイクルで命令#15がストールSTL1アサートによって実行されなかったため、レジスタ情報MR1は更新しない。そして、時刻t9同様バイパス制御BPE0A0，BPTB0A0及びBPTB0A1がアサートされる。また、セルSBTB2と読出し番号RA0がr0で一致し、スレッド番号THTB2とTH0が共に0で一致するのでバイパス制御BPTB2A0がアサートされる。更に、セルSBL3と読出し番号RB1がr2で一致し、スレッド同期番号IDL3とID1が共に0で一致するのでバイパス制御BPL3B1がアサートされる。また、時刻t9同様書き戻しBNE0,BNL0,BNL1及びBNTB0がネゲートされ、セルSBE0,SBL0,SBL1及びSBL2が更新され、書込み指示SE0及びSTB0 がネゲートされ、一時バッファ制御CE0及びCTB0がアサートされる。更に、セルSBL2及びSBTB1ではスレッド番号THL2及びTHTB1が1であるため図26の論理により書き戻しBNL2及びBNTB1は引き続きネゲートされる。また、セルSBL3及びSBTB2ではスレッド同期番号IDL3及びIDTB2とID0とが0で一致するため図26の論理により書き戻しBNL3及びBNTB2がアサートされる。そして、図27の第6及び第7式により書込み指示SL3及びSTB1はアサートされ、一時バッファ制御CL3及びCTB2はネゲートされる。この結果、図27の表により、時刻t8同様データ選択M0,M1及びM2はそれぞれE0,TB1及びTB2となり、これに従って一時バッファ制御情報SBTB0,SBTB1及びSBTB2が更新される。レジスタモジュールRMでも、時刻t8同様データ選択M0,M1及びM2に従って、一時バッファDTB0,DTB1及びDTB2が更新される。そして、書込み指示SL3及びSTB2によりレジスタファイルRFのレジスタr2及びr0にロードデータDL3及び一時バッファデータDTB2が書き戻される。また、バイパス制御BPE0A0,BPTB0A0,BPTB1A0及びBPTB2A0がアサートされたので、バイパスマルチプレクサMA0において、図30の論理により実行結果DE0が読出しデータDRA0として選択される。この時、バイパス制御BPTB0A0,BPTB1A0及びBPTB2A0によって一時バッファTBでは一時バッファ読出しデータTBA0として一時バッファデータDTB0が読出され、バイパスマルチプレクサMA0においてもBPTBA0がアサートされる。しかし、バイパス制御BPE0A0もアサートされているので、図30の論理により最も新しい実行結果DE0が選択される。更に、バイパス制御BPL3B1がアサートされたので、バイパスマルチプレクサMB1において、図3

0の論理によりロードデータDL3が読出しデータDRB1として選択される。また、読出しデータDRA1はレジスタファイルRFのレジスタr3から読出される。

時刻t12では、時刻t11と同様に、命令#9及び#15の命令アドレスステージA0及びA1、並びに、命令フェッチステージI0及びI1を行う。更に、時刻t10と同様に、命令#9及び#15の命令デコードステージD0及びD1、命令#9の命令実行ステージE0、並びに、命令#9のデータロードステージL1,L2及びL3を行う。そして、命令#15の実行ステージE1を行う。命令実行部EX1において、読出しデータDRA1とDRB1を加算し、実行結果DE1に出力する。

レジスタスコアボードRSは、時刻t12では図18のようになっている。時刻t11とスレッド同期番号が1ずつ減った以外はほぼ同一であるが、セルSBE1のレジスタr3への書込み情報が増えている。そして、セルSBE1と読出し番号RB0がr3で一致し、スレッド番号THE1とTH1が共に0で一致するのでバイパス制御BPE1A1がアサートされる。スコアボードの各セルは時刻t11と同様に更新される。レジスタモジュールRMでも、時刻t11と同様一時バッファTB及びレジスタファイルRFのレジスタr2及びr0が更新され、読出しデータDRA0及びDRB1が選択される。また、バイパス制御BPE1A1がアサートされたので、バイパスマルチプレクサMA1において、図30の論理により実行結果DE1が読出しデータDRA1として選択される。

時刻t13では、命令#9及び#15の命令アドレスステージA0及びA1を行う。命令供給部IF0は前サイクルと同様にリピート動作するが、リピート回数RC0が1なので、回数比較器CC0出力が１となり、ANDゲートが0となり、命令アドレスマルチプレクサMR0が命令#9のアドレス+4即ち命令#10の次の命令を指し、命令バッファの命令#9以降の命令の保持を解除する。リピート回数RC0をデクリメントして0とする。尚、命令#10の次命令以降の動作の説明は時刻t14以降では省略する。命令供給部IF1は時刻t9と同様にリピート動作しリピート回数RC0を4とする。時刻t12と同様に、命令#9及び#15の命令フェッチステージI0及びI1、命令デコードステージD0及びD1、命令実行ステージE0及びE1、並びに、命令#9のデータロードステージL1,L2及びL3を行う。

レジスタスコアボードRSは、時刻t13では図18のようになっている。時刻t12とスレッド同期番号が1ずつ減った以外は同一である。そして、時刻t12と同様に、ス

コアボードの各セルが更新され、レジスタモジュールRMの一時バッファTB及びレジスタファイルRFが更新され、読出しデータDRA0、DRA1及びDRB1が選択される。時刻t14では、時刻t13と同様に、命令#15の命令アドレスステージA1、命令フェッチステージI1、命令#9及び命令#15の命令デコードステージD0及びD1、命令実行ステージE0及びE1、並びに、命令#9のデータロードステージL1,L2及びL3を行う。また、リピート動作が解除されたので、命令#10を分岐系命令デコーダBDEC0でデコードし、SYNCE命令処理を行う。SYNCE命令はデータ使用スレッドの終了を待つ命令である。データ使用スレッド即ちスレッド1はリピート終了時にスレッド同期番号ID1が0に戻るので、スレッド同期番号ID0が0のままだと、データ使用スレッドは、データ定義スレッドを追い越さないというルールにより、止まってしまう。そこで、SYNCE命令デコードからデータ使用スレッド終了まではこのルールを無視するように命令マルチプレクサMX0及びMX1を制御する。この制御は命令#16から活用されるので図18では命令#16の命令アドレスステージA1として記載している。

レジスタスコアボードRSは、時刻t14では図18のようになっている。時刻t13とスレッド同期番号が1ずつ減った以外は同一である。そして、時刻t13と同様に、スコアボードの各セルが更新され、レジスタモジュールRMの一時バッファTB及びレジスタファイルRFが更新され、読出しデータDRA0、DRB1及びDRA1が選択される。

時刻t15では、時刻t14と同様に、命令#15の命令アドレスステージA1、命令フェッチステージI1、命令デコードステージD1、命令#9及び命令#15の命令実行ステージE0及びE1、並びに、命令#9のデータロードステージL1,L2及びL3を行う。

レジスタスコアボードRSは、時刻t15では図18のようになっている。時刻t14とはスレッド同期番号が1ずつ減り、RA0でのr0の読出しがなくなった以外は同一である。そして、時刻t14と同様に、スコアボードの各セルが更新される。但し、スコアボードセルSBE0及びSBL0には新たな書込み情報は保持されず、これらのセルは無効化される。また、レジスタモジュールRMの一時バッファTB及びレジスタファイルRFが更新され、読出しデータDRA1及びDRB1が選択される。

時刻t16では、時刻t15と同様に、命令#15の命令アドレスステージA1、命令フェッチステージI1、命令デコードステージD1、命令実行ステージE1、並びに、命令

#9のデータロードステージL1,L2及びL3を行う。命令アドレスステージA1では命令供給部IF1が前サイクルと同様にリピート動作するが、リピート回数RC1が1なので、回数比較器CC1出力が１となり、ANDゲートが0となり、命令アドレスマルチプレクサMR1が命令#15のアドレス+4即ち命令#17を指し、命令バッファの命令#15以降の命令の保持を解除する。リピート回数RC0をデクリメントして0とする。

レジスタスコアボードRSは、時刻t16では図18のようになっている。時刻t15とはスレッド同期番号が1ずつ減り、セルSBE0及びSBL0が無効になった以外は同一である。そして、時刻t15と同様に、スコアボードの各セルが更新される。但し、スコアボードセルSBL1及びSBTB0には新たな書込み情報は保持されず、これらのセルは無効化される。また、レジスタモジュールRMの一時バッファTB及びレジスタファイルRFが更新され、読出しデータDRA1及びDRB1が選択される。但し、レジスタr2への書込みは行われない。

時刻t17では、時刻t16と同様に、命令#15の命令フェッチステージI1、命令デコードステージD1、命令実行ステージE1、並びに、命令#9のデータロードステージL2及びL3を行う。

レジスタスコアボードRSは、時刻t17では図18のようになっている。時刻t16とはスレッド同期番号が1ずつ減り、セルSBL1及びSBTB0が無効になった以外は同一である。そして、時刻t16と同様に、スコアボードの各セルが更新される。但し、スコアボードセルSBL2及びSBTB1には新たな書込み情報は保持されず、これらのセルは無効化される。また、レジスタモジュールRMの一時バッファTB及びレジスタファイルRFが更新され、読出しデータDRA1及びDRB1が選択される。

時刻t18では、命令#16の命令フェッチステージI1を行う。命令供給部IF1は命令キューIQ1nの命令#16を命令I10として命令マルチプレクサMX1経由で命令デコーダDEC1に供給する。この時スレッド同期番号がデータ定義スレッドと同じ0となっているが、SYNCE命令によってデータ定義スレッド側はデータ使用スレッドの終了を待っており、同一スレッド同期番号の命令の発行が可能になっている。また、時刻t17と同様に、命令#15の命令デコードステージD1、命令実行ステージE1、並びに、命令#9のデータロードステージL3を行う。

レジスタスコアボードRSは、時刻t18では図18のようになっている。時刻t17とは

スレッド同期番号が1ずつ減り、セルSBL2及びSBTB1が無効になった以外は同一である。そして、時刻t17と同様に、スコアボードの各セルが更新される。但し、スコアボードセルSBL3及びSBTB2には新たな書込み情報は保持されず、これらのセルは無効化される。また、レジスタモジュールRMの一時バッファTB及びレジスタファイルRFが更新され、読出しデータDRA1及びDRB1が選択される。

時刻t19では、命令#16の命令デコードステージD1を行う。命令#16はレジスタr1の指すアドレスにレジスタr3の内容をストアする命令である。命令デコーダDEC1はこのための制御情報C1を出力する。また、レジスタ有効VR1のうち、VA1及びVB1をアサートする。また、時刻t17と同様に、命令#15の命令実行ステージE1を行う。また、命令供給部IF1の分岐系命令デコーダBDEC1が命令#17のTHRDEをデコードし、命令供給部IF1を停止させ、スレッド終了ETH1をアサートする。

レジスタスコアボードRSは、時刻t19では図18のようになっている。時刻t18とはスレッド同期番号が1ずつ減り、セルSBL3及びSBTB2が無効になり、レジスタ読出し番号RA1及びRB1が異なる以外は同一である。そして、時刻t18と同様に、スコアボードの各セルが更新される。但し、スコアボードセルSBE1には新たな書込み情報は保持されず、セルは無効化される。また、レジスタモジュールRMのレジスタファイルRFが更新される。但し、更新されるレジスタはr3のみである。また、読出しデータDRA1がレジスタファイルRFのr1から読み出される。更に、セルSBE1と読出し番号RB1のレジスタ番号がr3で一致し、スレッド番号THE1とTH1が一致するので、バイパス制御BPE1B1がアサートされ、読出しデータマルチプレクサMB1において実行結果DE1がDRB1として選択されて出力される。

時刻t20では、命令#16の命令実行ステージE1を行う。制御情報C1に従って読出しデータDRA1をストアアドレスとして実行結果DE1に出力し、読出しデータDRB1をデータとして実行結果DM1に出力する。

レジスタスコアボードRSは、時刻t19では図18のようになっている。時刻t18とはスレッド同期番号が1ずつ減り、セルSBL3及びSBTB2が無効になり、レジスタ読出し番号RA1及びRB1が異なる以外は同一である。そして、時刻t18と同様に、スコアボードの各セルが更新される。但し、スコアボードセルSBE1には新たな書込み情報は保持されず、セルは無効化される。また、レジスタモジュールRMのレジス

タファイルRFが更新される。但し、更新されるレジスタはr3のみである。また、読出しデータDRA1がレジスタファイルRFのr1から読み出される。更に、セルSBE1と読出し番号RB1のレジスタ番号がr3で一致し、スレッド番号THE1とTH1が一致するので、バイパス制御BPE1B1がアサートされ、読出しデータマルチプレクサMB1において実行結果DE1がDRB1として選択されて出力される。更に、スレッド終了ETHがアサートされたので、スコアボード制御CTLは図27の第5式により単一スレッドSTHをアサートする。

以上のように本実施例のマルチスレッド方式によりデータロード時間が隠蔽される。

本実施例ではデータ定義スレッドが定義してレジスタモジュールRMの一時バッファTBに書込んだデータをデータ使用スレッドが使用していない。データ使用スレッドが使用しているデータはロードデータでありロード直後に使用されて直接レジスタファイルRFに書込まれている。このように一時バッファが無駄に使用されていると、データロード時間を長くした場合に、更に無駄な書込みのためのバッファが必要になる。データロード時間が30ならば図16のプログラムを一時バッファフルSTLTBによるストールなしに実行するには29本の一時バッファが必要である。一時バッファのデータはバイパス制御で随時読出して命令実行部に供給する必要があるので本数を増やすことはハードウェア量の増大や実行速度の低下を招く。このような問題を避けるには、。データ定義スレッドが定義してデータ使用スレッドが使用するレジスタを限定すればよい。

例えば、リンクレジスタ指定命令で、特定のレジスタ又はレジスタグループをリンクレジスタに指定し、リンクレジスタのみをスレッド間データ転送に使用できると定義する。そして、図16のプログラムならばr2をリンクレジスタに指定する。すると、r2以外のレジスタはスレッド間の逆依存及び出力依存を考慮する必要がないので、実行結果を直接レジスタファイルRMに書込んでよい。すると、図18のパイプライン動作における一時バッファの使用は全てなくなる。

この時、データロード時間が30の場合に図16のプログラムをストールなしに実行するにはロードステージ数を30とし、L4からL29を追加すればよい。これに伴って、レジスタスコアボードにはSBL4からSBL29が追加する。そして、SBL0からSBL

28からのバイパス制御は全てストールSTL0及びSTL1にのみ反映し、データのバイパス経路は増加しない。

通常のプロセッサではデータロード時間は複数定義されている。オンチップキャッシュにヒットした場合、オンチップメモリにある場合、オフチップキャッシュにヒットした場合、あるいはオフチップメモリにある場合等である。例えばデータロード時間として、2,4,10及び30がありうる場合、SBL1,SBL3,SBL9及びSBL29に対応するバイパス経路を設け、データロード時間によって、ストールとバイパスを使い分けることにより、本発明を複数のデータロード時間に対応させることが可能である。また、本実施例では定義しなかったが除算命令のように実行時間の長い演算命令もある。このような命令用にデータロード用と同様なハードウェアを実現することは、本発明の属する技術分野の通常の技術者ならば可能である。

本実施例ではスレッド0及び1をデータ定義スレッド及びデータ使用スレッドに固定したが、前述のようにこの固定を外すことは本発明の属する技術分野の通常の技術者ならば可能である。そして、データ定義スレッドの処理終了後にTHRDE命令でこのスレッドを終了し、データ使用スレッドが新たにデータ定義スレッドとなってTHRDG命令で新たなスレッドを起動し、起動したスレッドを新たなデータ使用スレッドとするといったプログラムも可能となる。こうすると本実施例で用いたSYNCE命令が不要となり、スレッドが1本しかない時間を削減でき、性能が向上する。

また、本実施例ではデータの流れを一方通行に限定している。しかし、上記リンクレジスタ指定を行えは、双方向データ通信も可能である。各方向に対して異なるリンクレジスタを指定し、各スレッドでリンクレジスタのデータ定義命令の実行が終了したら、データ定義同期命令SYNCDを発行し、リンクレジスタの使用が終わったらデータ使用同期命令SYNCUを発行する。そして、SYNCU命令発行時にスレッド同期番号を更新する。SYNCU命令の代わりに本実施例のようなリピートによる同期を行っても良い。複数のスレッドで双方向のデータをやり取りする例は、データ依存関係が少ないけれどもないわけではないといったルーズカップリングな処理を同時実行する場合に有効である。図31にスレッド間双方向データ通信

方式のプログラム処理の流れを示す。

まず、リンクレジスタ指定命令RNCRで、スレッドTH0からTH1へはr2を、逆方向にはr3をリンクレジスタに指定する。そして、スレッドTH0及びTH1でそれぞれリンクレジスタ定義命令#01及び#11を実行する。その後、データ定義同期命令SYNCDを発行し、リンクレジスタ使用命令#0t及び#1yを実行する。そして、最後にデータ使用同期命令SYNCUを発行する。それぞれのスレッドの実行時間は変動する可能性がある。スレッドTH0に比べてスレッドTH1の実行が早い場合、図のTH1.aのようになる。この時、スレッドTH0データ定義同期命令SYNCD発行をスレッドTH1のリンクレジスタ使用命令#1yが待つので、フロー依存関係の検出ミスは起こらない。また、スレッドTH1の実行が遅い場合、図のTH1.bのようになる。この時、スレッドTH1データ定義同期命令SYNCD発行をスレッドTH0のリンクレジスタ使用命令#1tが待つので、フロー依存関係の検出ミスは起こらない。データ定義同期命令SYNCDはスレッド間の実行優先度を変更している。但し、この例における実行優先度はリンクレジスタ毎に異なっている。r2に関してはスレッドTH0がTH1に、r3に関してはスレッドTH1がTH0に優先する。

本実施例ではレジスタを介してスレッド間データ通信を行っているが、レジスタ番号の代わりにメモリアドレスの全部又は一部を使用してメモリを管理することにより、メモリを介してスレッド間データ通信を行うようにすることは本発明の属する技術分野の通常の技術者ならば可能である。

【０００６】

【発明の効果】

本発明により、大規模なアウトオブオーダ実行やソフトウェアパイプラインニングを行った場合と同等な性能を、従来のマルチスレッドプロセッサに簡単な制御機構を追加することによりより単純で少ないハードウェアにより達成することができる。また、従来のマルチスレッドプロセッサが多数のスレッドの同時又は時分割実行をしなければ達成できない性能を、2個程度の少ないスレッド数で達成することができる。そして、スレッド数が少ない分スレッド生成及び終了のオーバヘッドを削減することができ、多数のスレッドの状態を記憶するためのハードウェアも削減できる。

【図面の簡単な説明】

【図１】

サンプルプログラム。

【図２】

リピート制御命令。

【図３】

２並列スーパスカラプロセッサのパイプライン例。

【図４】

図１のプログラムのロードレイテンシ４の２並列スーパスカラパイプライン動作。

【図５】

図１のプログラムのロードレイテンシ４の２並列スーパスカラアウトオブオーダパイプライン動作。

【図６】

ソフトウェアパイプラインにより図１のプログラムのロードレイテンシ４を隠蔽した例。

【図７】

図６のプログラムのロードレイテンシ４の２並列スーパスカラパイプライン動作。

【図８】

図１のプログラムをMerlot方式の４並列マルチプロセッサ用に書き換えた例。

【図９】

図８のプログラムのロードレイテンシ４のパイプライン動作。

【図10】

図１のプログラムを特開平8-249183のマルチスレッドプロセッサ用に書き換えた例。

【図11】

図10のプログラムのロードレイテンシ４のパイプライン動作。

【図12】

既存方式の所要サイクル数比較。

【図13】

本発明及び従来方式によるスレッド分割方式。

【図14】

従来方式によるロードレイテンシ８の場合のスレッド実行。

【図15】

本発明によるロードレイテンシ８の場合のスレッド実行。

【図16】

本発明のマルチスレッドによりロードレイテンシ４を隠蔽した例。

【図17】

２並列マルチスレッドプロセッサのパイプライン例。

【図18】

図16のプログラムのロードレイテンシ４のパイプライン動作。

【図19】

本発明を適用した２スレッドプロセッサの例。

【図20】

命令供給部の例。

【図21】

命令選択部の例。

【図22】

命令マルチプレクサによる命令選択組合せ。

【図23】

レジスタスコアボードの構成例。

【図24】

ロード系セル入力マルチプレクサの例。

【図25】

スコアボード先頭セルの例。

【図26】

スコアボード非先頭セルの例。

【図27】

スコアボード制御論理の例。

【図28】

レジスタモジュールの例。

【図29】

一時バッファの例。

【図30】

バイパスマルチプレクサの例。

【図31】

スレッド間双方向データ通信方式の例。

【書類名】図面

【図１】

図１

サンプルプログラム

```
Cソース
N=8;
for (i=0; i<N;i++)
    y += x[i];
```

```
#1.                 LDRS _repeat_start
#2.                 LDRE _repeat_end
#3.                 LDRC #8
#4.                 MOV #x_addr, r0
#5.                 MOV #y_addr, r1
#6.                 MOV #0, r3
#7. _repeat_start   MOV @r0+, r2
#8. _repeat_end     ADD r2, r3
#9.                 MOV r3, @r1
```

【図２】

図２

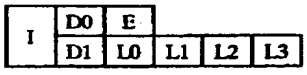
リピート制御命令

```
LDRS _repeat_start : リピート開始PCを_repeat_startとする
LDRE _repeat_end   : リピート終了PCを_repeat_endとする
LDRC #N            : リピート回数をNとする
```

【図３】

図３

２並列スーパスカラプロセッサのパイプライン例



【図４】

図４

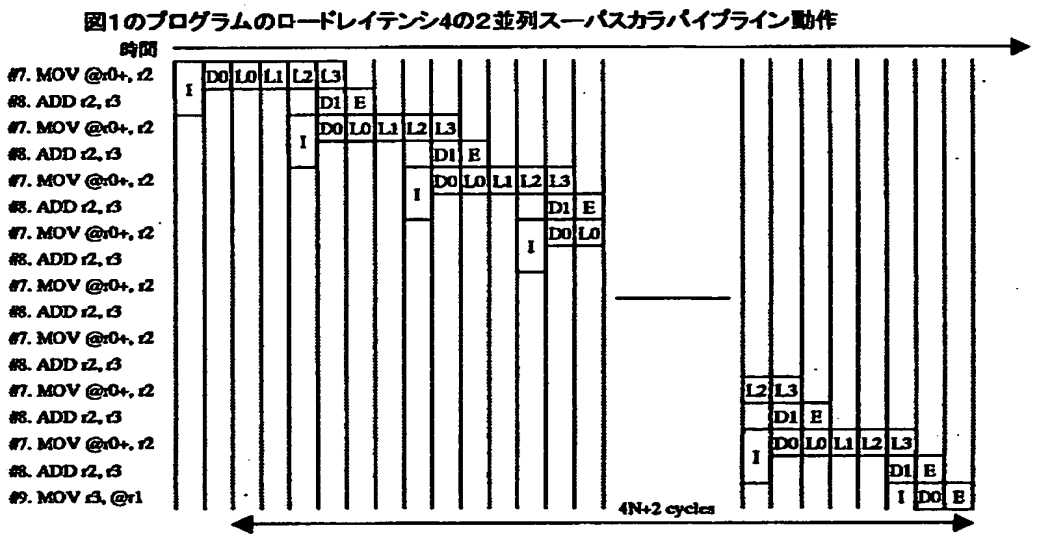
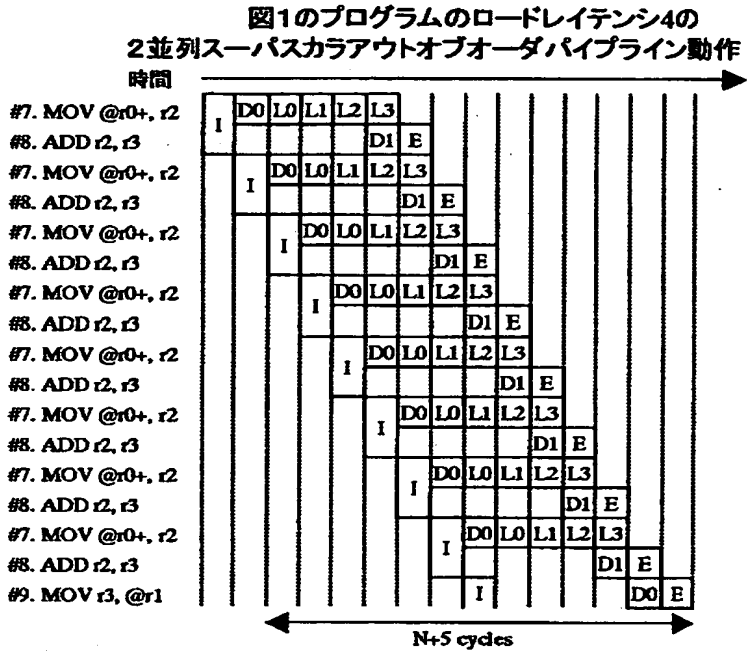


図1のプログラムのロードレイテンシ4の2並列スーパスカラパイプライン動作
時間
#7. MOV @r0+, r2
#8. ADD r2, r3
#9. MOV r3, @r1
I
D0
L0
L1
L2
L3
D1
E
4N+2 cycles

【図５】

図５



【図６】

図６

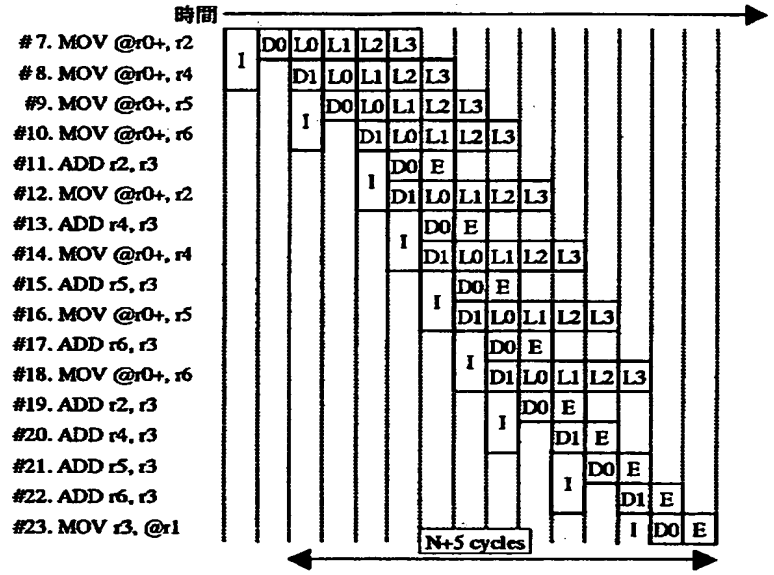
ソフトウェアパイプラインにより図1のプログラムのロードレイテンシ4を隠蔽した例

```
#1. LDRS _repeat_start      #7. MOV @r0+, r2      #11.  _repeat_start  ADD r2, r3      #19. ADD r2, r3
#2. LDRE _repeat_end        #8. MOV @r0+, r4      #12.                 MOV @r0+, r2    #20. ADD r4, r3
#3. LDRC #2                 #9. MOV @r0+, r5      #13.                 ADD r4, r3      #21. ADD r5, r3
#4. MOV #x_addr, r0         #10. MOV @r0+, r6     #14.                 MOV @r0+, r4    #22. ADD r6, r3
#5. MOV #y_addr, r1                               #15.                 ADD r5, r3      #23. MOV r3, @r1
#6. MOV #0, r3                                    #16.                 MOV @r0+, r5
                                                  #17.                 ADD r6, r3
                                                  #18.  _repeat_end    MOV @r0+, r6
```

【図７】

図７

図6のプログラムのロードレイテンシ4の2並列スーパスカラパイプライン動作



【図８】

図８

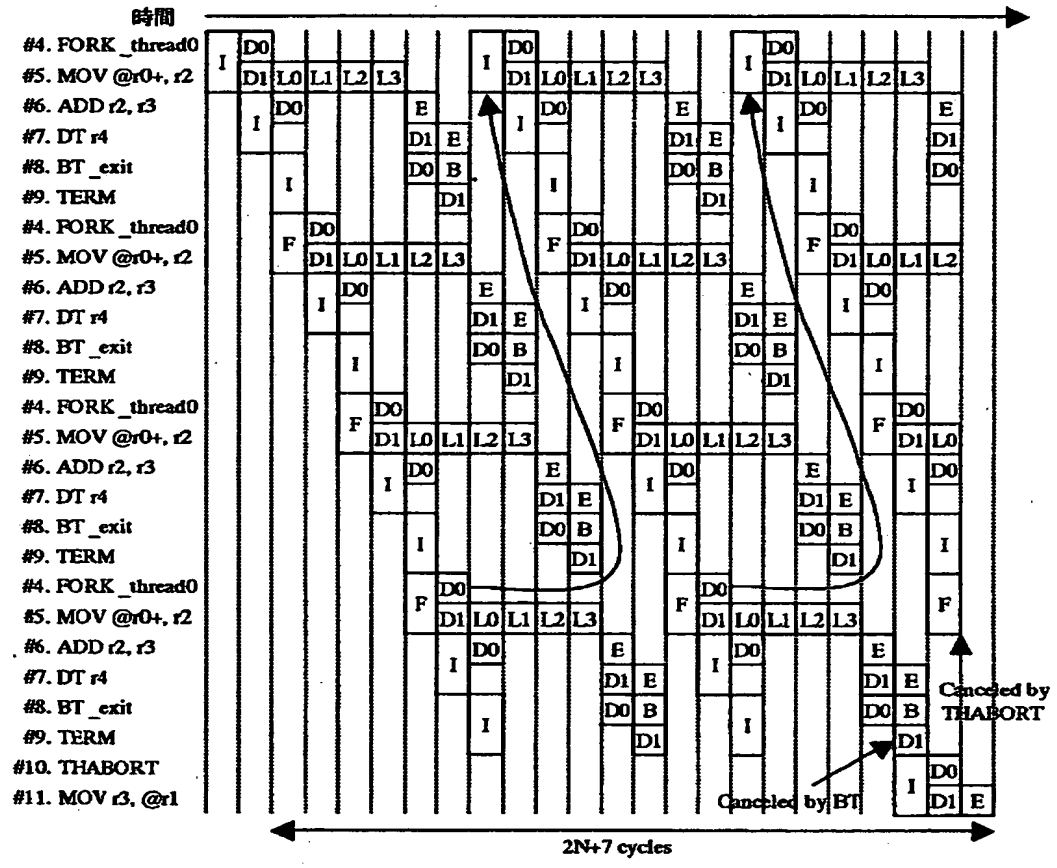
図1のプログラムをMerlot方式の4並列マルチプロセッサ用に書き換えた例

#1.		MOV #8,r4	#5.	MOV @r0+, r2	#9.		TERM
#2.		MOV #x_addr, r0	#6.	ADD r2, r3	#10.	_exit	THABORT
#3.		MOV #y_addr, r1	#7.	DT r4	#11.		MOV r3, @r1
#4.	_thread0	FORK _thread0	#8.	BT/S _exit			

【図９】

図９

図8のプログラムのロードレイテンシ4のパイプライン動作



【図１０】

図１０

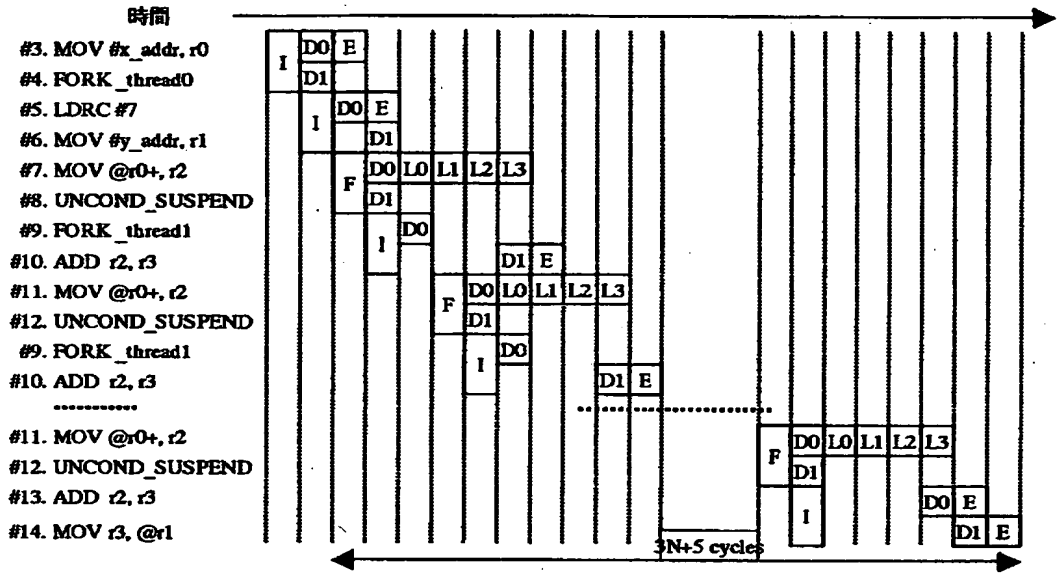
図1のプログラムを特開平8-249183のマルチスレッドプロセッサ用に書き換えた例

```
#1. LDRS _repeat_start      #7. _thread0     MOV @r0+, r2
#2. LDRE _repeat_end        #8.              UNCOND_SUSPEND
#3. MOV #x_addr, r0         #9. _repeat_start FORK _thread1
#4. FORK _ thread0          #10.             ADD r2, r3
#5. LDRC #7                 #11. _thread1    MOV @r0+, r2
#6. MOV #y_addr, r1         #12. _repeat_end UNCOND_SUSPEND
                            #13.             ADD r2, r3
                            #14.             MOV r3, @r1
```

【図１１】

図１１

図10のプログラムのロードレイテンシ4のパイプライン動作



【図１２】

図１２

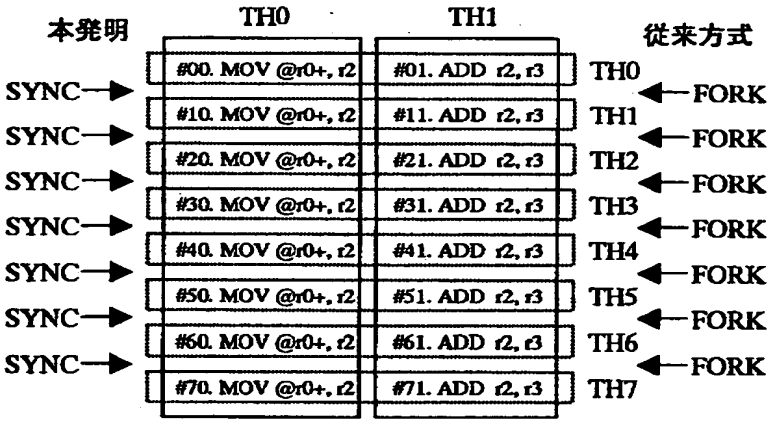
既存方式の所要サイクル数比較

	データ数	ロードレイテンシ	アウトオブオーダ、ソフトウェアパイプライン	Merlot 方式	特開平 8-249183	通常プロセッサ
#1	N	L	N+L+1	MAX(2N+L+2, (L+3)N/4+7)	MAX(3N+L+1, (L-1)N+5)	LN+2
#2		4	N+5	2N+7	3N+5	4N+2
#3		30	N+31	33N/4+7	29N+5	30N+2
#4	8	4	13	23	29	34
#5		30	39	73	237	242
#6	32	4	37	51	101	130
#7		30	63	271	933	962

【図１３】

図１３

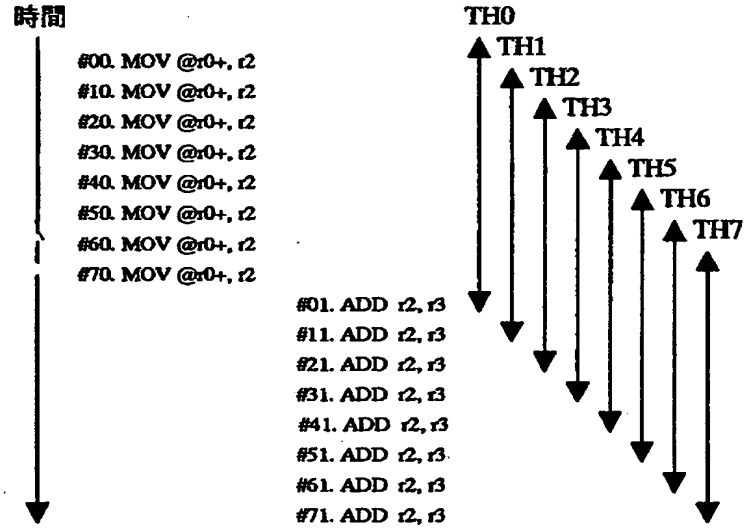
本発明及び従来方式によるスレッド分割方式



【図１４】

図１４

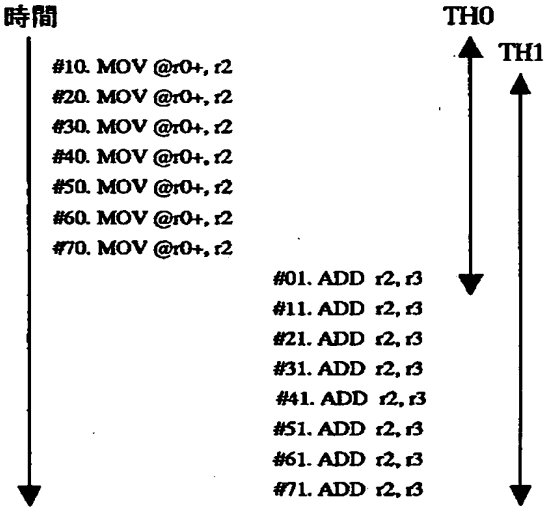
従来方式によるロードレイテンシ8の場合のスレッド実行



【図１５】

図１５

本発明によるロードレイテンシ8の場合のスレッド実行



【図１６】

図１６

本発明のマルチスレッド方式によりデータロード時間を隠蔽した例

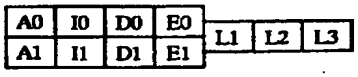
#1.	LDRE _repeat0	#11. _thred1	LDRS _repeat1
#2.	MOV #x_addr, r0	#12.	LDRE _repeat1
#3.	LDRS _repeat0	#13.	LDRC #8
#4.	MOV #y_addr, r1	#14.	NOP
#5.	THRDG/R _thred1	#15. _repeat1	ADD r2, r3
#6.	MOV #0, r3	#16.	MOV r3, @r1
#7.	LDRC #8	#17	THRDE
#8.	NOP		
#9. _repeat0	MOV @r0+, r2		
#10.	SYNCE		

THRDG/R: スレッド生成 REPEATタイプ，THRDE: スレッド終了，SYNCE: スレッド終了同期

【図１７】

図１７

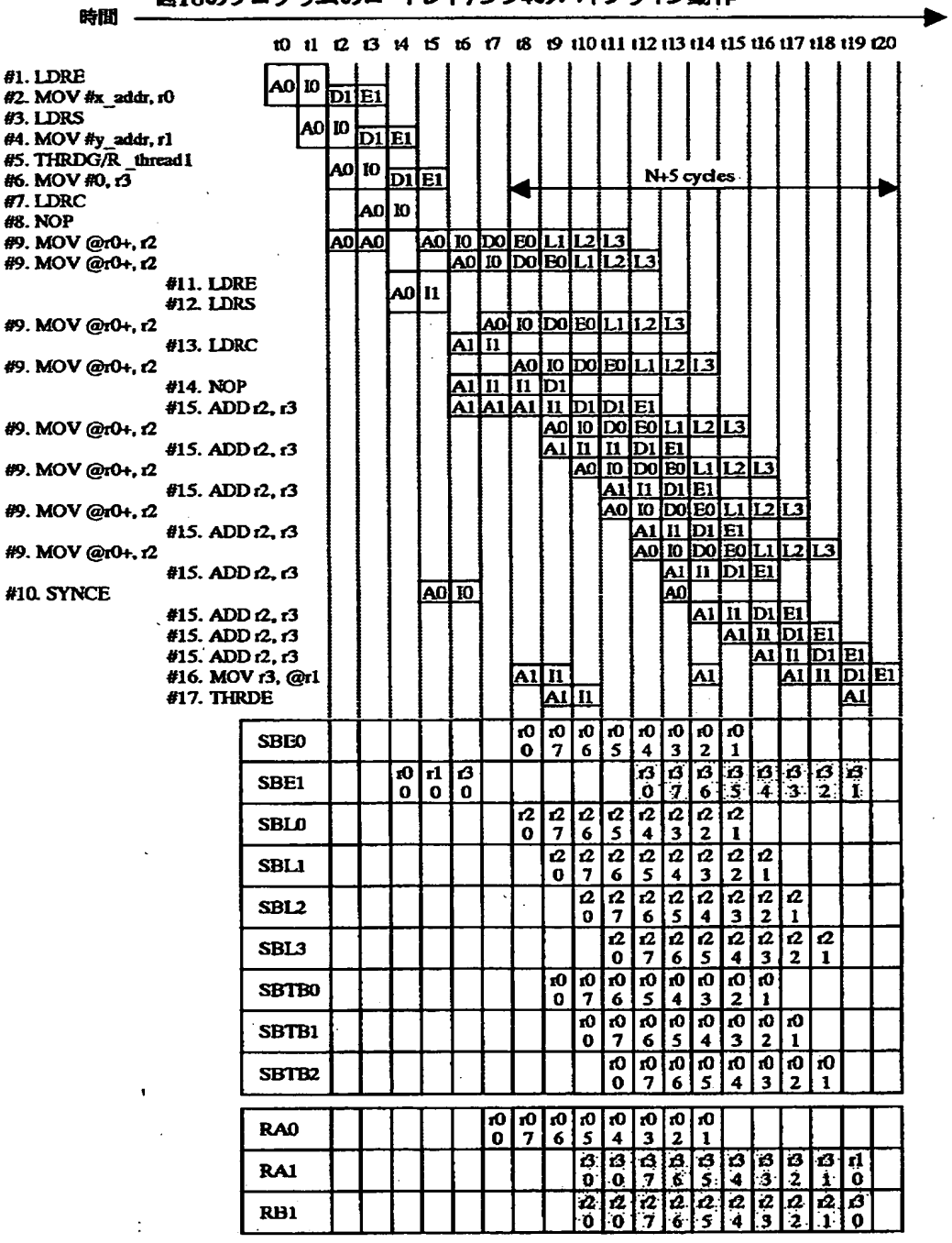
2並列マルチスレッドプロセッサのパイプライン例



【図１８】

図１８

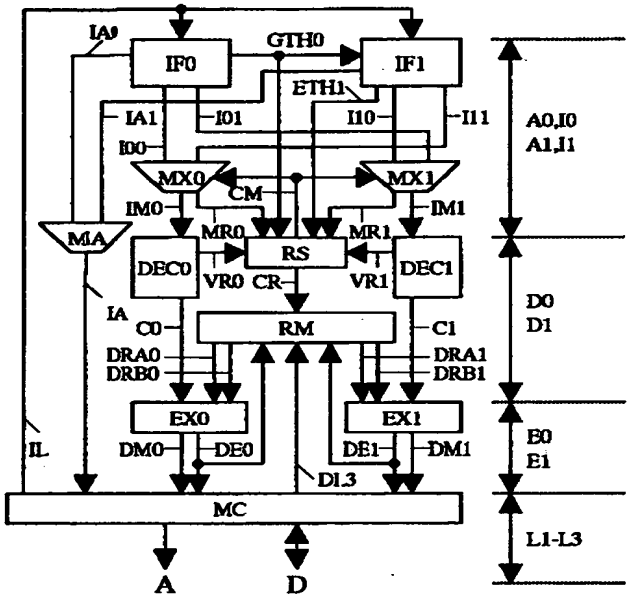
図16のプログラムのロードレイテンシ4のパイプライン動作



【図１９】

図１９

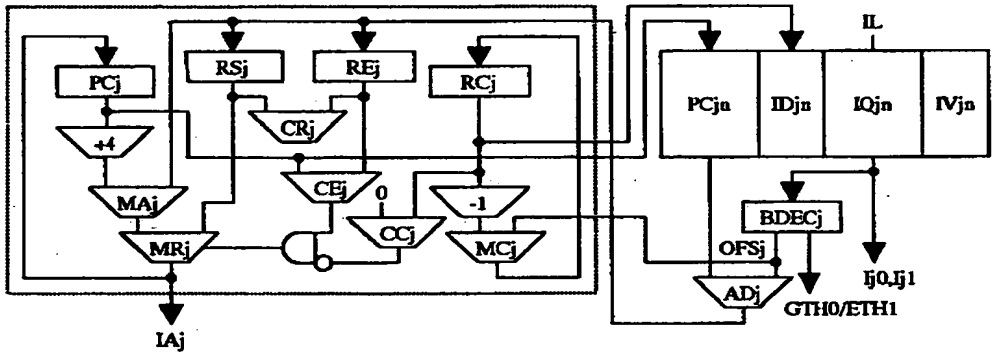
本発明を適用した2スレッドプロセッサの例



【図２０】

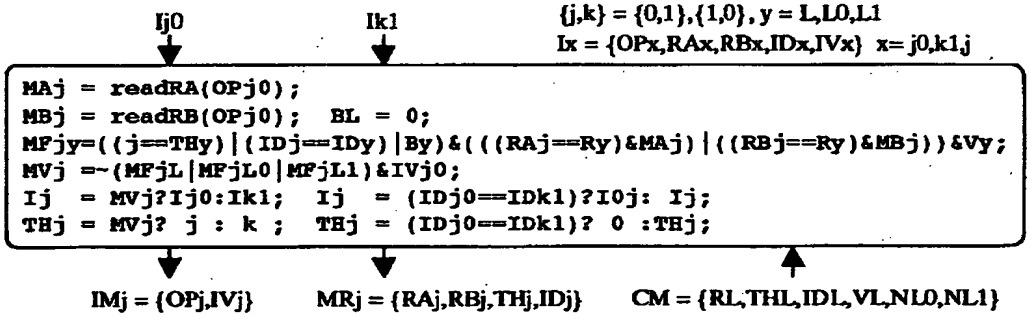
図２０

命令供給部の例



【図２１】

図２１
命令選択部の例



【図２２】

図２２
命令マルチプレクサによる命令選択組合せ

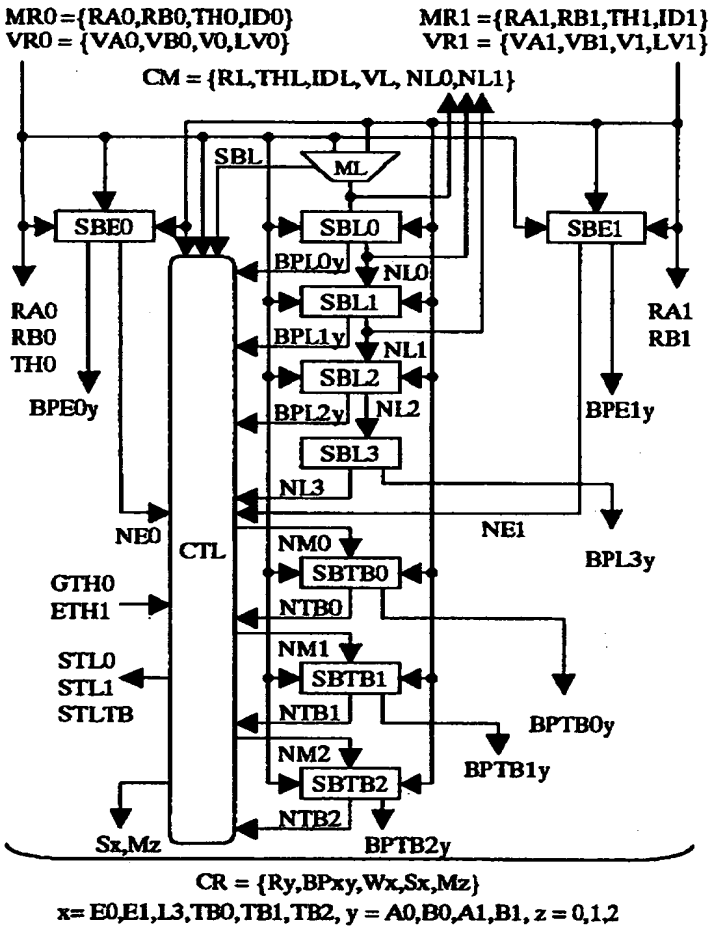
	実行可否		選択命令	
	I00	I10	I0	I1
#1	可	可	I00	I10
#2	可	否	I00	I01
#3	否	可	I11	I10
#4	否	否	I11	I01

出証特２００１－３１０６８０６

【図２３】

図２３

レジスタスコアボードの構成例



【図２４】

図２４

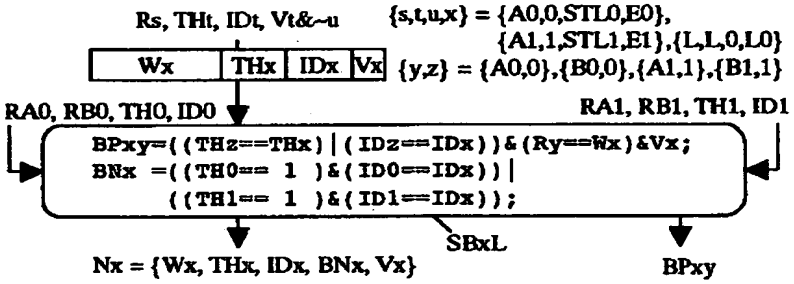
ロード系セル入力マルチプレクサの例

```
SBL = ((TH0==0) & LV0) | ((TH0==1) & ~LV1);
if (SBL) {RL = RB0; THL = TH0; IDL = ID0; VL = LV0&~STL0}
else     {RL = RB1; THL = TH1; IDL = ID1; VL = LV1&~STL1}
```

【図２５】

図２５

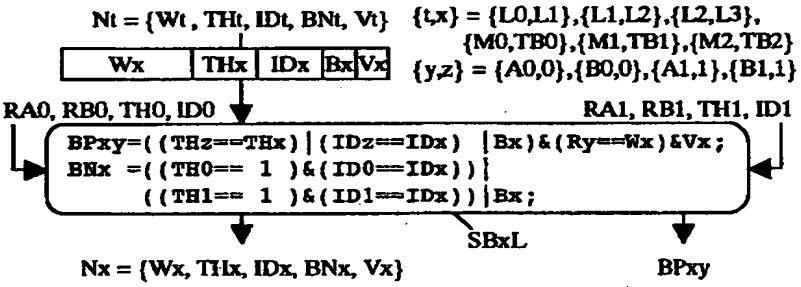
スコアボード先頭セルの例



【図２６】

図２６

スコアボード非先頭セルの例



【図２７】

図２７

スコアボード制御論理の例

```
STL0=((BPL0A0 | BPL1A0 | BPL2A0)& VA0)|
     ((BPL0B0 | BPL1B0 | BPL2B0)& VB0);
STL1=((BPL0A1 | BPL1A1 | BPL2A1)& VA1)|
     ((BPL0B1 | BPL1B1 | BPL2B1)& VB1);
STL0 |= STL1 & (TH0==1)|((SBL==1)&LV0);
STL1 |= STL0 & (TH0==0)|((SBL==0)&LV1);
STH= (~GTH0 & STH)| ETH1;
Sx = Vx & ((THx==1)| Bx | STH)          (x = TB0,TB1,TB2,L3,E0,E1)
Cx = Vx & ((THx==0)&~Bx &~STH)
```

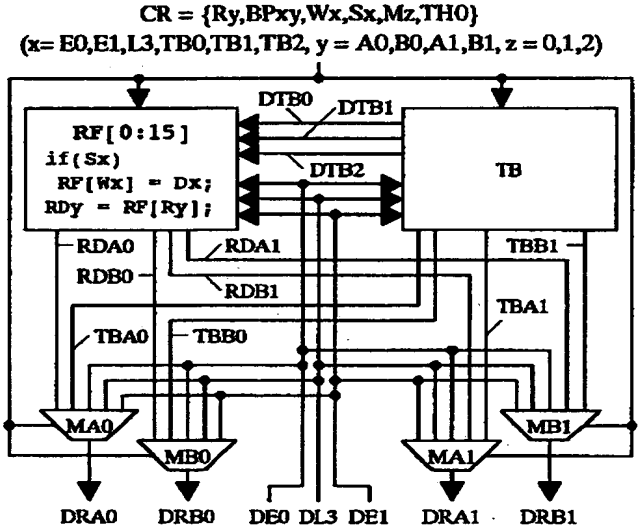
状態						出力			
CTB2	CTB1	CTB0	CL3	CE0	CE1	M2	M1	M0	STLTB
*	*	*	0	0	0	TB2	TB1	TB0	0
*	0	*	0	0	1	TB1	TB0	E1	0
0	*	*							
*	0	*	0	1	0	TB1	TB0	E0	0
0	*	*							
*	0	*	1	0	0	TB1	TB0	L3	0
0	*	*							
0	0	*	0	1	1	TB0	E0	E1	0
			1	0	1	TB0	L3	E1	0
			1	1	0	TB0	L3	E0	0
0	0	0	1	1	1	L3	E0	E1	0
その他						TB2	TB1	TB0	1

```
NM2=(M2==TB2)?NTB2:((M2==TB1)?NTB1:((M2==TB0)?NTB0:((M2==L3)?NL3)));
NM1=(M1==TB1)?NTB1:((M1==TB0)?NTB0:((M1==L3 )?NL3 :((M1==E0)?NE0)));
NM0=(M0==TB0)?NTB0:((M0==L3 )?NL3 :((M0==E0 )?NE0 :((M0==E1)?NE1)));
```

【図２８】

図２８

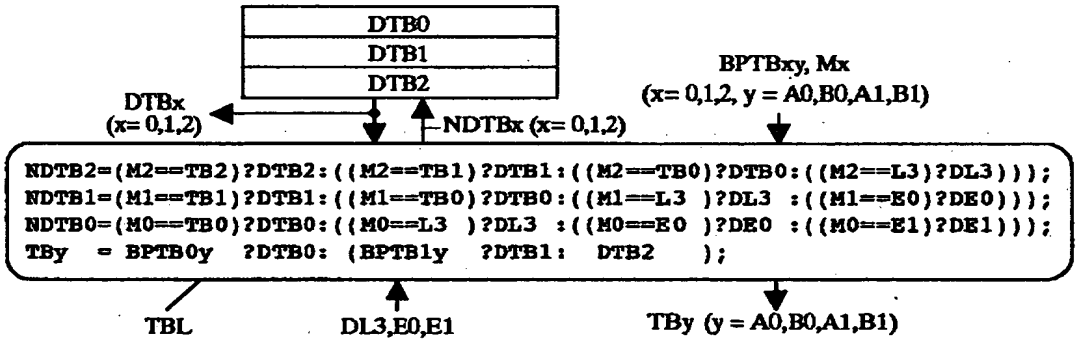
レジスタモジュールの例



【図２９】

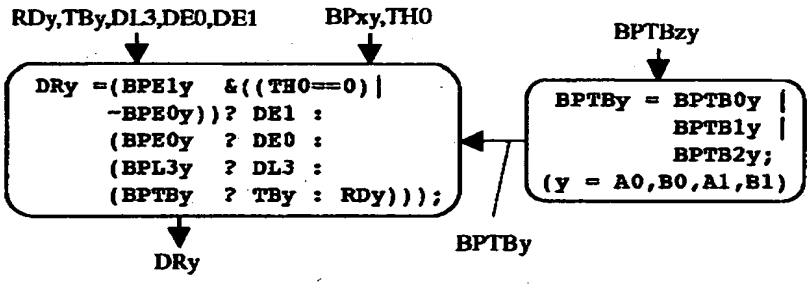
図２９

一時バッファの例



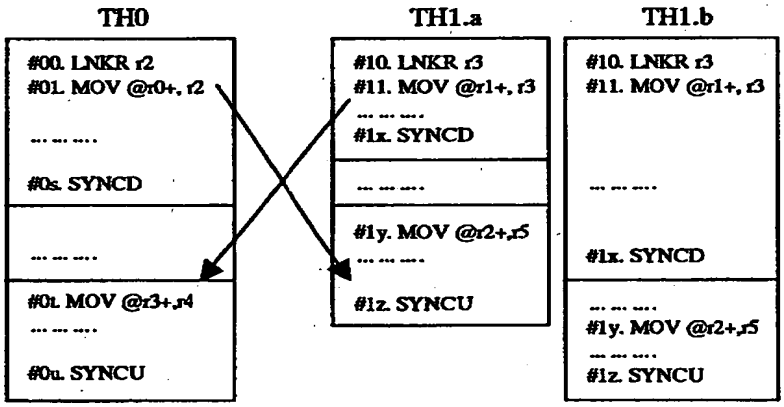
【図３０】

図３０
バイパスマルチプレクサの例



【図３１】

図３１
スレッド間双方向データ通信方式の例



【書類名】要約書

【要約】

【課題】本発明が解決しようとする課題は、大規模なハードウェア追加や根本的なアーキテクチャ変更を行わずに、少ないハードウェアの追加で単一プロセッサで複数スレッドを実行するマルチスレッドプロセッサを改良することにより、数十命令規模の並列度抽出を可能にし、性能を向上させることである。

【解決手段】マルチスレッドプロセッサに、スレッド間優先度の時分割変更機能を追加し、スレッド間のデータの流れを限定して流れる順に実行順序を規定することにより、データ依存関係のある複数スレッドを同時又は時分割実行し、大規模なアウトオブオーダ実行並の性能を達成した。

【選択図】図19

認定・付加情報

特許出願の番号	特願２００１－０６２７９２
受付番号	５０１００３１７６１０
書類名	特許願
担当官	第七担当上席　　００９６
作成日	平成１３年　３月　８日

＜認定情報・付加情報＞

【提出日】　平成13年 3月 7日

次頁無

出　願　人　履　歴　情　報

識別番号　　〔０００００５１０８〕

１．変更年月日　　１９９０年　８月３１日
［変更理由］　新規登録
住　所　東京都千代田区神田駿河台４丁目６番地
氏　名　株式会社日立製作所